Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
Chipsmall
采购部

AVR® ATICE30

User Guide

Table of Contents

Section 5

Section 6

Section 7

Section 8

Section 9

Section 10

Section 1

Introduction

1.1 General Description

The Atmel AVR® ATICE30 is an in-circuit emulator for all ATmega devices. The ICE30 is controlled by AVR Studio, version 3.0 and later. The ICE30 currently supports the following AVR devices:

- ATmega103
- ATmega603
- ATmega161
- ATmega163
- ATmega83
- ATmega32

This document describes the configuration and setup of the ICE30 and configuration of AVR Studio to support emulation of these devices. The ICE30 consists of the following components:

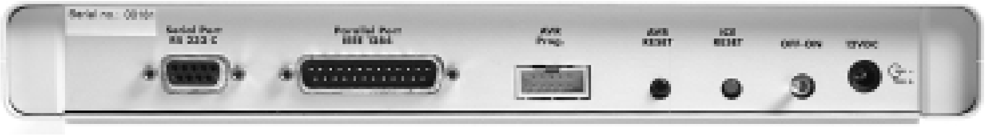
- ICE30 emulator unit
- Pod card ATmegaPOD
- Pod card AT90ADCPOD
- Pod card ATmega163POD
- Pod cable
- ATmega161/ATmega163/ATmega83/ATmega32 probe (DIL40) with cable
- ATmega103/ATmega603 probe with TQFP64 adapter, including cables
- RS-232 cable
- Power supply unit
- Documentation

The ICE30 emulator unit is the main part of the AVR ICE30 system. The ICE unit is controlled by AVR Studio, which runs on a host PC. The AVR Studio automatically detects if there is an emulator present on one of the PC's serial ports. Note that if no ICE is detected, AVR Studio will show **Simulator** in the lower right corner of the main window. If the ICE30 is detected, **AVR Emulator** will be indicated and your connections are correct. AVR Studio will issue a warning if a program previously run in emulator mode is started in simulator mode.

1.2 External Connections

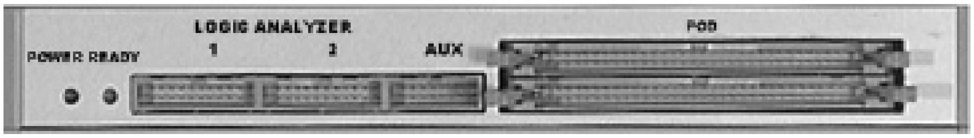
The ICE unit is connected to an RS-232 port on the host PC with the supplied RS-232 cable. The connection on the back panel is shown in Figure 1-1. The **Parallel Port** and **AVR Prog.** connectors have no function on this ICE. Two reset buttons are placed on the back panel of the ICE unit. The **AVR RESET** button resets the application. The **ICE RESET** button resets both the ICE unit and the application.

Figure 1-1. ICE Unit Back Panel



The ICE unit front panel is shown in Figure 1-2. Two LEDs on the front panel indicate the status of the emulator. After power-up, the red LED will be lit, indicating that the power supply is OK and the green LED is turned on after a few seconds when initialization and self-test are finished.

Figure 1-2. ICE Unit Front Panel



The emulator pods are connected to the **POD** connector on the ICE unit. The **LOGIC ANALYZER** and **AUX** connectors are described later. See “Connector Description” in Section 9.

1.3 Power System

The ICE30 system has an internal power regulator that delivers 15W at 5V. The ICE30 itself uses about 10W. The power supply delivered with the ICE30 is dimensioned to meet the requirements of the emulator. If another power supply is used, it should supply a voltage between 9 and $15V_{DC}$, and a minimum of 20W. The battery eliminator connector on the AVR ICE30 system is a standard type with 2.1 mm center tap. Ground should be connected to the center tap.

The target application can be powered either by the emulator or by its own power supply when using pods AT90ADCPOD or ATmegaPOD. When using ATmega163POD, the emulator cannot power the target application.

Note: Turn off power to the target application before the emulator is turned off or you may cause damage to the pod.

1.4 Reset System

The ICE30 has two independent reset systems. One is for the ICE30 itself and the other is for the emulated AVR device. The ICE30 reset button is placed on the back panel of the box. The button is labeled **ICE RESET** and is hidden in the back panel for safety reasons. If the emulator starts to behave unpredictably, use a thin tool to push this reset button. The green LED will be turned off for a while and will be switched on again when the system is ready. When the ICE30 reset button is pressed, the program memory is cleared, thus the project file must be closed in AVR Studio and then reopened.

The AVR reset system can reset the emulated device both when the device is running and stopped. This reset can be activated from several sources:

- The push-button marked **AVR RESET** on the back panel of the ICE unit (only when running)
- The push-button marked **RESET** on the pod (only when running)
- The reset button in AVR Studio. Note that the reset button in AVR Studio will stop the emulation process if it is running when the button is pushed.
- A reset button placed in the user application (only when running)

Please note that only a reset from the user application will reset other components in the application connected to the AVR's reset pin.

1.5 Trace Buffer

The AVR ATmegaICE has a 32K x 96-bit trace buffer that stores information about program execution for every clock cycle. When the emulator is stopped, this trace buffer can be examined to extract information about the history of the emulated program. The details on which data are stored and how to retrieve them are described in the "AVR Studio User Guide." When the trace buffer is full, it will wrap around and start overwriting the oldest entries.

The trace buffer can be turned on or off at any program line. This makes it possible to skip tracing delay loops and other subroutines which would otherwise fill the trace memory with unnecessary data. The trace buffer is inactive by default. To trace an entire program, a **Trace on** marker should be placed on the first line of the program in AVR Studio.

1.6 External Triggers

The AVR ATmegaICE has five external trigger inputs and five trigger outputs, all located on the Aux connector next to the Pod connector.

- The trigger inputs can act as break signals to the emulator and/or they can be logged in the trace buffer. Any inputs set up to break the emulator are activated when a rising edge is detected.
- The trigger outputs may be set as trigger points on any instruction in the code window in AVR Studio. If enabled on an instruction, the output(s) will remain high for one AVR clock cycle when the marked instruction is executed. This can be used to trigger a logic analyzer or an oscilloscope.

The details on how to enable and set up triggers and mask registers are presented in **AVR Studio Help**. There are three global mask registers that are used to control the behavior of the triggers:

- The Trigger Output Global Mask Register controls which of the output pins are allowed to be controlled by the trigger settings in the code. An output pin that is disabled will remain low even if a trigger point for that particular pin is set in the code.
- The Trigger Input Global Mask Register controls which of the input pins are allowed to break the emulator. If more than one line is enabled, the emulator will break on either one, but will not store any information about which input caused the event. Note that unconnected inputs are pulled high by internal pull-up resistors. Unused lines must not be enabled. Otherwise, the emulator may break on the first line.
- The External Trace Mask Register controls which of the input pins will be stored in the trace memory. Input pins that are not enabled in this register will be stored as zero in the trace memory. To be traced, input signals must be valid and stable at the rising edge of the AVR clock and for 50 ns thereafter. It is also necessary that the trace buffer is enabled and turned on.

The trigger input and the external trace are two independent functions acting on the same input pins. Note that the trigger logic is asynchronous and edge driven, whereas the trace logic is clocked on the AVR clock. The emulator may therefore break on a glitch signal that is too narrow to be traced.

1.7 Preparing the ICE30 System for Use

Complete the following procedure in order to start using the ICE30. Before connecting the probe cable to the user application:

- Connect the RS-232 cable between the ICE30 unit and the PC serial port.
- Connect the desired pod card to the ICE30 unit with the supplied pod cable.
- Connect the probe cable to the pod.
- Connect the enclosed power supply (9 - 15V_{DC}) to the ICE30 unit.
- Turn on the power and check that the red LED marked **POWER** is lit.
- After a short time (<10 s), the green LED marked **READY** will be lit and the ICE30 system will be ready.
- Turn off the power.
- If an SMD adapter is used, solder this to the target PCB.
- Plug the probe into the application/adapter, paying attention to connect it correctly. If it is not connected correctly, the ICE30 system may be damaged.
- Turn on the power. Wait for the green LED to be lit.
- Connect power to the target application.
- Start AVR Studio.
- Make sure that the jumpers and AVR Studio settings are set according to the requirements.

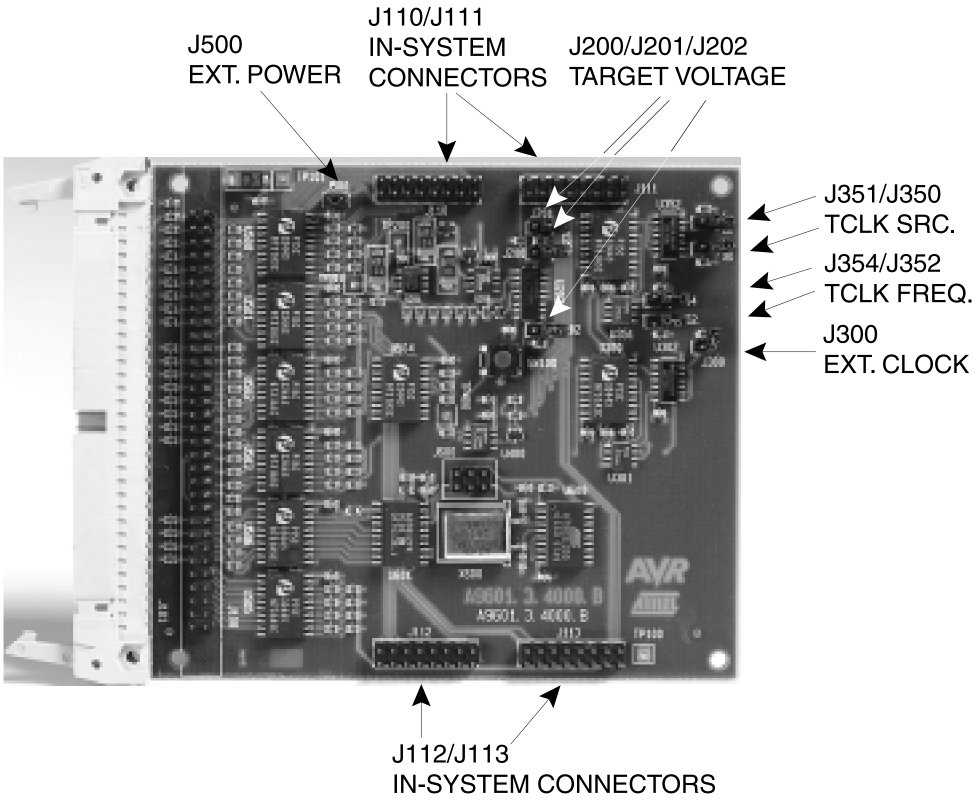
Section 2

Emulating ATmega103/603

2.1 The ATmegaPOD

Use ATmegaPOD to emulate ATmega103/603. The ATmegaPOD contains several jumpers which must be set to achieve the desired operation. The jumpers involved in the configuration are indicated in Figure 2-1.

Figure 2-1. ATmegaPOD Jumper Placements



2.2 Configuration of the ATmegaPOD

2.2.1 Voltage Regulation for Port Pins

If the target system uses its own power supply, the output voltage must be set to the voltage in the target system. The jumpers J200 - J202 are used for this purpose. The interpretation of the different settings of these jumpers is shown in Figure 2-2.

Figure 2-2. Jumper Settings

Target V_{CC} (V)	J202	J201	J200
2,7 - 2,9	▮	▮	▮
3,0 - 3,3	▮	▮	:
3,4 - 3,7	▮	:	▮
3,8 - 4,1	▮	:	:
4,2 - 4,5	:	▮	▮
4,6 - 4,8	:	▮	:
4,9 - 5,1	:	:	▮
5,2 - 5,5	:	:	:

2.2.2 Power to Target PCB

The ATmegaICE can power the target PCB with 5.0V. To enable this option, J500 should be mounted. Note that the absolute maximum supply current is 1.0A, so the target should use less than 5W of power to use this option. Jumpers J200 - J202 must all be removed when external power is selected. If the target application uses voltages other than 5.0V, it must supply its own power and J500 must be removed.

2.2.3 Clock Setting

The external clock system can be set to use either an external crystal or an external clock signal. To use an external crystal, select **External Oscillator** in the **Emulator Options** menu in AVR Studio, and select an appropriate frequency range for the mounted crystal. It is important to tune the oscillator driver by choosing the range in the **Clock Range** menu to make the clock system work properly with an external crystal. J300 must be open when using an external crystal. If an external clock signal is used (on pin XTAL1), jumper J300 must be mounted. This setting also requires the user to select **External Oscillator** in the **Emulator Options** menu in AVR Studio.

ATmegaICE also has a very accurate programmable internal clock that can be used for emulation. Select this clock by choosing **Internal Oscillator** in the **Emulator Options** menu in AVR Studio. The setting of J300 on the ATmegaPOD and crystals and clock signals in the user application are insignificant when the internal oscillator is selected.

2.2.4 Timer Oscillator Setting

In order to enable the timer oscillator, jumper J351 must be mounted. When J351 is mounted, the setting of J350 determines whether to use an external clock signal or the local oscillator. If J350 is mounted, the external clock signal is used (on pin TOSC1). If J350 is not mounted, an external crystal is expected to be mounted between the TOSC1 and TOSC2 pins. When an external crystal is used (J350 not mounted), jumpers J352 and J354 should be used to indicate the frequency range of the oscillator. The interpretation of the different settings for these jumpers is given in Figure 2-3.

Figure 2-3. Jumper Settings

Frequency Range	J354	J352
10 kHz - 100 kHz	:	:
100 kHz - 1 MHz	:	▮
1 MHz - 5 MHz	▮	:
5 MHz - 10 MHz	▮	▮

As noted above, the selected frequency range may not correspond to the crystal frequency due to load capacitors and/or stray capacitors. Due to long probe leads, it may be difficult to emulate the 32 kHz oscillator in many designs. A way to work around this is to lift pins 2 and 3 on U351 from the PCB and solder a 32 kHz crystal directly to the pins, with neither the pins nor the crystal touching the PCB.

2.3 Connecting to the System

The pod card is connected to ICE30 with the pod cable (the wide cable). Do not disassemble the pod cable. Connect the pod to the target application via the 4-probe cables and the TQFP probe adapter.

The device’s pins are placed on the four connectors, J110, J111, J112 and J113. The pinout of these connectors is shown in Table 2-1, Table 2-2, Table 2-3 and Table 2-4. The **#** column shows the pin number on each header. The **&** column shows the corresponding pin number on an ATmega103/ATmega603 device. The numbering of the pins on each header is shown in Figure 2-5.

It is possible to use the ATmegaPOD without the TQFP64 adapter (Figure 2-4) if four 8 x 2-pin headers are added to the target application PCB. Table 2-1, Table 2-2, Table 2-3 and Table 2-4 give details on how to use these header connectors. Connect the pin headers on the target application to the connectors on ATmegaPOD by using the probe cables included in your kit.

Figure 2-4. ATmega103/603 Probe

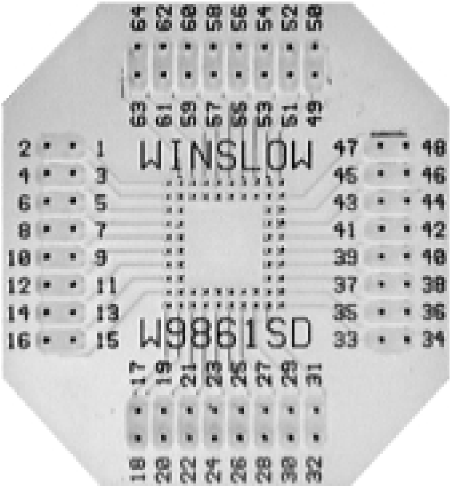


Table 2-1. Pinout for Header J110

J110						
#		&		#		&
1	PE0 (PDI/RXD)	2		2	$\overline{\text{PEN}}$	1
3	PE2 (AC+)	4		4	PE1 (PDO/TXD)	3
5	PE4 (INT4)	6		6	PE3 (AC-)	5
7	PE6 (INT6)	8		8	PE5 (INT5)	7
9	PB0 ($\overline{\text{SS}}$)	10		10	PE7 (INT7)	9
11	PB2 (MOSI)	12		12	PB1 (SCK)	11
13	PB4 (OC0)	14		14	PB3 (MISO)	13
15	PB6 (OC1B)	16		16	PB5 (OC1A/PWM1A)	15

Table 2-2. Pinout for Header J111

J111						
#		&		#		&
1	TOSC2	18		2	PB7 (OC2/PWM2)	17
3	$\overline{\text{RESET}}$	20		4	TOSC1	19
5	GND	22		6	VCC	21
7	XTAL1	24		8	XTAL2	23
9	PD1 ($\overline{\text{INT1}}$)	26		10	PD0 ($\overline{\text{INT0}}$)	25
11	PD3 ($\overline{\text{INT3}}$)	28		12	PD2 ($\overline{\text{INT2}}$)	27
13	PD5	30		14	PD4	29
15	PD7 (T2)	32		16	PD6 (T1)	31

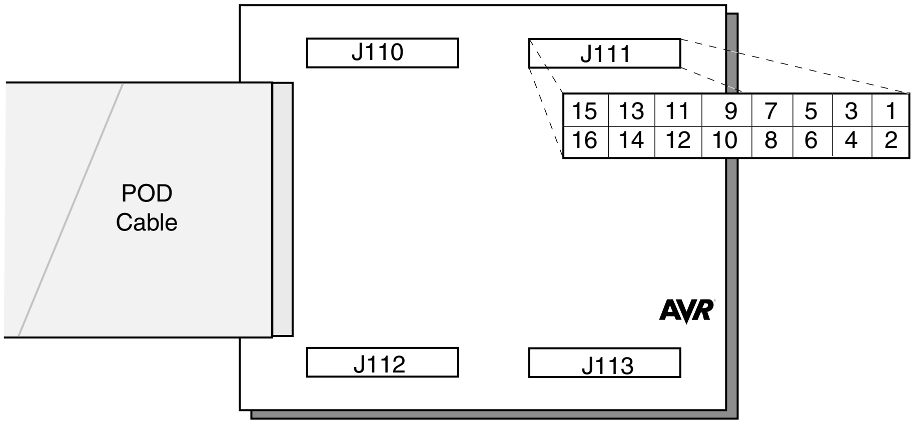
Table 2-3. Pinout for Header J112

J112						
#		&		#		&
1	$\overline{\text{RD}}$	34		2	$\overline{\text{WR}}$	33
3	PC1 (A9)	36		4	PC0 (A8)	35
5	PC3 (A11)	38		6	PC2 (A10)	37
7	PC5 (A13)	40		8	PC4 (A12)	39
9	PC7 (A15)	42		10	PC6 (A14)	41
11	PA7 (AD7)	44		12	ALE	43
13	PA5 (AD5)	46		14	PA6 (AD6)	45
15	PA3 (AD3)	48		16	PA4 (AD4)	47

Table 2-4. Pinout for Header J113

J113						
#		**&**		**#**		**&**
1	PA1 (AD1)	50		2	PA2 (AD2)	49
3	VCC	52		4	PA0 (AD0)	51
5	PF7 (ADC7)	54		6	GND	53
7	PF5 (ADC5)	56		8	PF6 (ADC6)	55
9	PF3 (ADC3)	58		10	PF4 (ADC4)	57
11	PF1 (ADC1)	60		12	PF2 (ADC2)	59
13	AREF	62		14	PF0 (ADC0)	61
15	AVCC	64		16	AGND	63

Figure 2-5. Pin Numbering



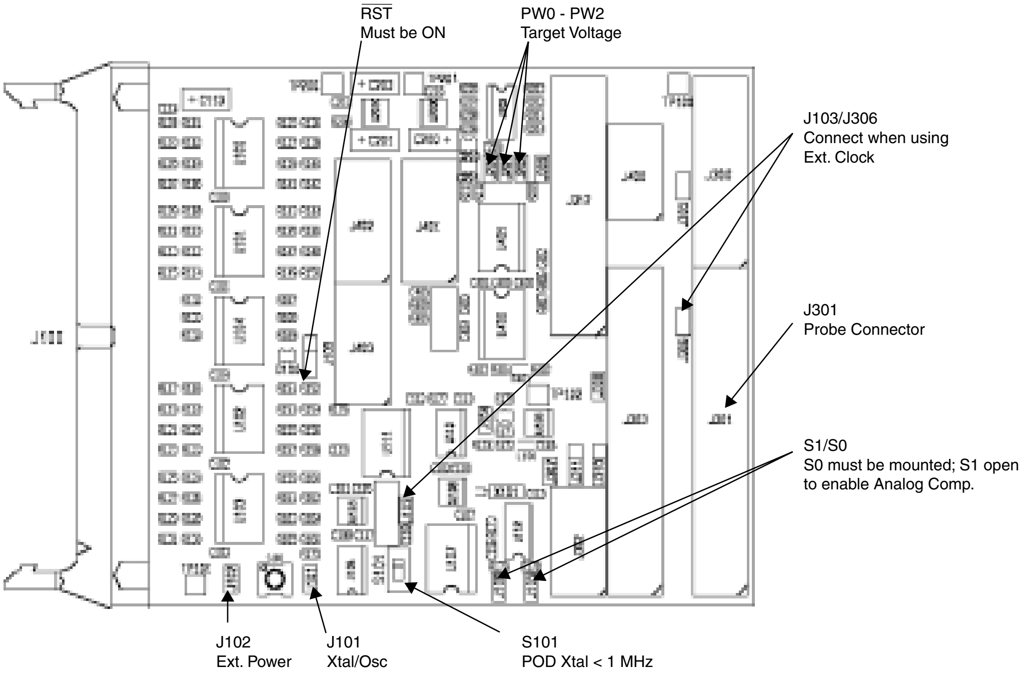
Section 3

Emulating ATmega161

3.1 The ATADCPOD

Use AT90ADCPOD to emulate ATmega161. To configure the pod several jumpers must be set. The jumpers involved in the configuration are indicated in Figure 3-1.

Figure 3-1. AT90ADCPOD Jumper Placements



3.2 Configuration of the AT90ADCPOD

3.2.1 Voltage Regulation for Port Pins

If the target system uses its own power supply, the jumper named **Ext. Power** (J102) must, under all circumstances, be removed and the output voltage must be set to the voltage in the target system. The jumpers PW0 - PW2 are used for this purpose. The interpretation of the different settings of these jumpers is shown in Figure 3-2.

Figure 3-2. Jumper Settings

Target V_{CC} (V)	PW2 (J202)	PW1 (J201)	PW0 (J200)
2,7 - 2,9	▮	▮	▮
3,0 - 3,3	▮	▮	:
3,4 - 3,7	▮	:	▮
3,8 - 4,1	▮	:	:
4,2 - 4,5	:	▮	▮
4,6 - 4,8	:	▮	:
4,9 - 5,1	:	:	▮
5,2 - 5,5	:	:	:

3.2.2 Power to Target PCB

The ATmegaICE can power the target application with 5.0V. To enable this option the jumper named **Ext. Power** should be mounted. Jumpers PW0 - PW2 must all be removed when external power is selected. Note that the absolute maximum supply current is 1.0A, so the target should use less than 5W of power to use this option. If the target application uses voltages other than 5.0V, it must supply its own power and J500 must be removed.

3.2.3 Clock Setting

AT90ADCPOD can use one of three available clock sources: the programmable internal clock in the ICE30, a crystal or an external oscillator in the user application.

The internal clock can be selected by choosing **Internal Oscillator** in the **Emulator Options** menu in AVR Studio. The internal clock can be adjusted between 400 kHz and 20 MHz.

To use an external clock source, select **External Oscillator** in the **Emulator Options** menu in AVR Studio. This clock signal can be in the range of 32.768 kHz to 8 MHz. It is important to tune the oscillator driver by choosing the range in the **Clock Range** menu to make the clock system work properly with an external crystal. On the pod card, the XTAL pins are connected to the ICE by using the 2-wire cable. Connect J103 to J306 when emulating ATmega161. Pin 1 on each connector is labeled with *****.

If the clock source from the user application is a crystal, connector J101 must be left open. If the clock source from the user application is an oscillator, a jumper must be mounted on connector J101. The switch S101 must be in the **OFF** position in both cases.

Long leads from your external crystal to the oscillator circuit on the pod may cause problems. It is possible to mount a crystal in the socket near J103. Do not use the 2-wire cable if you choose this option. If the crystal is above 1 MHz and this is selected in the **Clock Range** menu in **Emulator Options** in AVR Studio, S101 should be in the **ON** position. S101 should be in the **OFF** position under all other circumstances.

3.2.4 The Analog Comparator

To make the analog comparator work properly when emulating ATmega161, jumper S0 (J106) must be mounted and jumper S1 (J105) must be open.

3.2.5 $\overline{\text{RST}}$ Connector

The $\overline{\text{RST}}$ connector must have its jumper placed in the **ON** position.

The connectors not described in this document are intended for use with other devices/emulators.

3.3 Connecting to the System

The pod card is connected to the bottom pod connector of ICE30 using the pod cable (the wide cable). The 40-pin probe cable (DIL) should be connected to the S8515 probe connector (J304) and to the target application.

Note: It is important that the probe cable is correctly connected to the user application. The colored wire of the probe cable indicates pin 1 of the AVR device.

Section 4

Emulating ATmega163/ATmega83

4.1 The ATmega163POD

To emulate ATmega163/83/32, the ATmega163POD should be used. The ATmega163POD is fully configurable from AVR Studio. It includes two sockets that can be used to supply clock signals to the target application.

A crystal can be mounted in the socket labeled **ECLK** (external clock) to supply system clock to the target application. Both ATmega163/83/32 and ATmega163POD have an asynchronous timer that can be clocked by an external crystal. To emulate this, a crystal can be mounted in the socket labeled **TCLK** (timer clock).

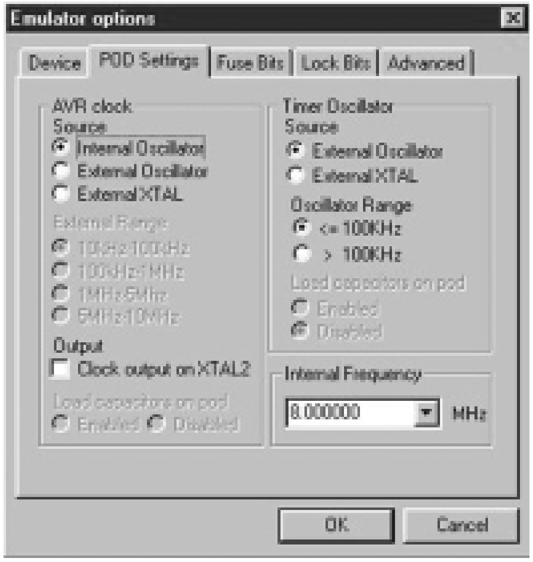
See below for detailed explanations of the ATmega163POD clock options and configurations.

The ATmega163POD has a reset button, and this can be used to reset the application when it is running.

4.2 Configuration of the ATmega163POD

ATmega163POD is configured directly from AVR Studio. When an object file is opened in AVR Studio for the first time, a dialog box with the ICE30 emulator options is displayed (Figure 4-1). The option can also be changed later from the **Options > Emulator Options** menu.

Figure 4-1. Emulator Options Dialog



AVR clock

Source

● Internal Oscillator

○ External Oscillator

○ External XTAL

External Range

● 10kHz-100kHz

○ 100kHz-1MHz

○ 1MHz-5MHz

○ 5MHz-10MHz

Output

□ Clock output on XTAL2

Load capacitors on pod

○ Enabled ○ Disabled

Timer Oscillator

Source

● External Oscillator

○ External XTAL

Oscillator Range

● <= 100KHz

○ > 100KHz

Load capacitors on pod

○ Enabled

● Disabled

Internal Frequency

8.000000 MHz

4.2.1 AVR Clock Source

- **Internal Oscillator:** If a clock is not available in the target application, select internal oscillator. The internal frequency is selectable in the range 400 kHz to 8 MHz. Note that this option is not available in an actual device. If the internal RC oscillator in ATmega163/83 is used, the frequency will be fixed to 1 MHz.
- **External Oscillator:** If the target application supplies its own clock, select external oscillator as clock source.
- **External XTAL:** If a crystal in the target application or in the socket on the pod card is used, select external XTAL as clock source. A crystal should not be mounted in the socket on the pod card if there are any connections on the XTAL pins in the target application.

If internal oscillator is selected, any signals on XTAL pins are overridden.

4.2.2 AVR Clock External Range

If an external clock source or a crystal is selected, the clock range must be specified. Please select from the four ranges available.

Note: Note that the selected frequency range may not correspond to the crystal frequency due to load capacitor and/or stray capacitors in the system.

This selection is not available if internal oscillator is selected.

4.2.3 AVR Clock Output

If internal oscillator is selected, this box can be checked to output the AVR clock on the XTAL2 pin.

4.2.4 AVR Clock Load Capacitors

If a crystal is mounted in the socket on the pod card it may be desirable to connect the load capacitors on the XTAL lines. The load capacitors have the value 22 pF and can be enabled or disabled here.

4.2.5 Timer Oscillator Source

The timer oscillator can have one of three different clock sources:

- **External Oscillator:** If the target application uses an external oscillator.
- **External XTAL:** If the target application uses a crystal or the crystal socket on the pod card is used.
- **No Clock:** If the timer oscillator is not used, a clock source does not have to be supplied.

4.2.6 Timer Oscillator Range

The emulator needs to know the clock speed of the timer oscillator clock. Please select from the two ranges available.

4.2.7 Timer Oscillator Load Capacitors

If a crystal is mounted in the socket on the pod card, load capacitors may be connected to the TOSC lines. The load capacitors have the value 22 pF and can be enabled or disabled here.

Note: The timer oscillator settings are only active when the timer oscillator is used.

4.2.8 Internal Frequency

Selects the frequency of the AVR clock. This setting is available only if internal oscillator is selected as AVR clock source. The internal frequency is selectable in the range 400 kHz to 8 MHz.

4.3 Connecting to the System

The pod card is connected to the bottom pod connector of ICE30 using the pod cable (the wide cable). The 40-pin probe cable (DIL) should be connected to the probe connector and to the target application.

Note: It is important that the probe cable is correctly connected to the user application. The colored wire of the probe cable indicates pin 1 of the AVR device.

Section 5

AVR Studio Emulator Options

When opening a new project the **Emulator Options** dialog will appear. This dialog can also be found in AVR Studio under **Options > Emulator Options**.

5.1 Device

Select the device from the list. The device list includes all devices currently supported in the emulator.

The pod cables may introduce noise problems on the ALE line when emulating 103/603 and 161 with external memory. As a workaround, ATmega103/603 and ATmega161 can be emulated with 64 KB internal RAM. Alternatively, an apx. 68 pF capacitor may be hooked to the ALE line.

5.2 Clock Source (ATmega103, ATmega603 and ATmega161)

If a crystal in the target application or in the socket on the pod card is used, select external oscillator as clock source. The clock range for the external oscillator must be specified. Please select from the four ranges available. This selection is not available if internal oscillator is selected.

If a clock is not available in the target application, select internal oscillator. The internal frequency is selectable in the range 400 kHz to 8 MHz. Note that this option is not available in an actual device.

The internal RC oscillator option of the ATmega163/ATmega83/ATmega32 device can be emulated by selecting internal oscillator and setting the frequency to 1 MHz.

5.3 Fuse Bits (ATmega163, ATmega161, ATmega83 and ATmega32)

The only fuse settings that affect the emulator's operation is the BOOTRST fuse and the BOOTSIZE fuses. In ATmega163/161/83/32 it is possible to read the fuse settings by means of LPM (and SPMCR). Therefore, all fuses are possible to set from AVR Studio even if they do not affect the emulator's behavior. The only way to program/clear the fuses is via this **Options** window. In an actual device, the fuses will be programmed/cleared when programming the Flash memory.

5.4 Lock Bits (ATmega163, ATmega161, ATmega83 and ATmega32)

The BLB0 and BLB1 lock bits determine whether or not LPM and SPM should be allowed within the different program memory blocks. See the ATmega163/161/83/32 datasheet for details. The LB lock bits do not affect the emulator's behavior. In ATmega163/161/83/32, it is possible to read and program the lock bits by means of LPM and SPM. Therefore, all lock bits are possible to set from AVR Studio. The only way to clear the lock bits is to clear them from this **Options** window. In an actual device, the lock bits will be cleared when performing a chip-erase.

5.5 Advanced (ATmega163, ATmega161, ATmega83 and ATmega32)

In order to emulate the enhanced instruction set featured by the enhanced AVR architecture (ATmega163/161/83/32), the option **Enable Enhanced Instruction Set** must be checked.

Note: When emulating the enhanced instruction set, emulation will no longer be in real time. See Section 6 “Emulator Limitations” for a detailed description.

Section 6

Emulator Limitations

6.1 Enhanced AVR Architecture

ATmega163/83, ATmega161 and ATmega32 use an enhanced AVR architecture supporting new features and instructions. The following features have been added in the new architecture.

- A powerful 2-cycle hardware multiplier supporting both signed/unsigned multiplication and fractional format.
- The XRAM interface has been given a longer hold time, thereby meeting the timing requirements of earlier unsupported external memory devices.
- Extended LPM/ELPM instructions that now support both post-increment and the possibility to select destination register.
- Self-programming capabilities. The program memory can be reprogrammed by the MCU itself.
- Move Word (MOVW) instruction enabling one-cycle 16-bits register copy.

The only device supporting the new XRAM interface is ATmega161. The XRAM access will internally take one clock cycle longer in the emulator compared to the actual device. For the accessed XRAM, the timing will be identical to that of an actual ATmega161 device.

The AVR ICE30 is built around the standard AVR core. Therefore, it does not directly support the Enhanced Instruction Set of the AVR enhanced architecture. As a workaround, these instructions are implemented in software in the emulator microcontroller.

To enable the Enhanced Instruction Set emulation, select the option **Enable Enhanced Instruction Set** in the **Options > Emulator Options** dialog in AVR Studio.

If enabled, the emulator will stop when reaching an Enhanced Instruction Set instruction, and the emulator MCU will perform the actions needed to emulate the instruction. This will typically be calculations and writing to the involved registers. This will take substantially longer time compared to the time used by the instruction in an actual AVR device. Table 6-1 shows how long the emulation will be stopped when emulating the different instructions. These emulation times will be independent of the target frequency.

If the BOOTRST fuse is programmed (0), there will also be an emulation break when the emulator reaches address 0x0000. This will happen both after a reset, after a jump to address 0x0000 and after a wraparound from the last instruction in the program memory.

Table 6-1. Enhanced Instruction Emulation Break Time

Instruction	Emulation Break Time [ms]
MOVW	3,4
LPM[1]	3,6
LPM Rd,Z	3,6
LPM Rd,Z+	3,8
ELPM[2]	4,6
ELPM Rd,Z	4,6
ELPM Rd,Z+	4,9
MUL	5,8
MULS	5,8
MULSU	5,8
FMUL	5,8
FMULS	5,8
FMULSU	5,8
SPM (SPMCR=0x01)[3]	5,7
SPM (SPMCR=0x03)[3]	11,5
SPM (SPMCR=0x05)[3]	26,8
SPM (SPMCR=0x09)[3]	2,4
JMP to Addr 0x0000	0,8

Notes:
1. Only if the **Enable Enhanced Instruction Set** option is selected.
2. Only if BOOTRST fuse is programmed. Will also occur immediately after a reset.
3. If SPM is executed from the application section, or the lock bits are programmed and therefore SPM is ignored, the emulation break time will be approximately 1.1 ms.

When emulating these instructions, the emulator IO clock is stopped. This means that any ongoing UART, SPI or I2C transmission/reception or PWM will be frozen, and therefore produce erroneous signals. To avoid this, the program should wait for any transmission/reception or PWM to finish before the enhanced instruction is executed.

Some applications have time-critical tasks where these emulation breaks are not acceptable. If this is so, the program should not use any of the Enhanced Instruction Set instructions shown in Table 6-2. Nor should the BOOTRST fuse be programmed if the program during execution jumps to addr 0x0000.

In the IAR C-compiler an option can be set whether the code should be compiled for the standard or the enhanced AVR architecture. This option can be found in **Project > Options > General > Target > Processor Configuration > Enhanced Core**.

Table 6-2. Enhanced Instruction Set

Enhanced Instruction Set
MOVW
LPM Rd,Z
LPM Rd,Z+
ELPM Rd,Z
ELPM Rd,Z+
MUL
MULS
MULSU
FMUL
FMULS
FMULSU
SPM

Note that LPM and ELPM with no operands are not parts of the Enhanced Instruction Set and will be emulated in real time as long as the Enhanced Instruction Set is not enabled. In the enhanced AVR architecture the lock and fuse bits may be read using LPM/ELPM. To achieve this in ICE30, the **Enable Enhanced Instruction Set** option must be enabled.

ATmega103/603 does not support the Enhanced Instruction Set.

If ATmega103/603 is emulated and the code contains Enhanced Instruction Set instructions, emulation will break and a warning box will appear.