Device support in IAR Embedded Workbench for 8051®

This guide describes how you can add support for a new device to IAR Embedded Workbench and how you can modify the characteristics of an already supported device.

Introduction

The information in this guide covers the most common properties of an 8051 device. If the device at hand has features which are not covered by this guide, you should contact IAR Systems for advice on how to proceed. This guide is only applicable to scenarios where hardware debugging is used. Full use of the simulator requires additional information about a device.

Adding support for a new device

To add support for a new device, fill out the form EW8051_DeviceCharacteristics.xls which you can find in the directory `8051\doc`. When finished, e-mail the form to **info@iar.com** and make sure to write "*To the product manager for Embedded Workbench for 8051"* in the subject field. Based on this information, IAR Systems will, if the information is complete, generate the set of device files required for integration in IAR Embedded Workbench.

Using the form, you must provide information about:

- Device
- Code banking
- Data pointer
- Multiple DPTR
- Memory layout
- SFR information.

Device

General information which is mandatory:

- **Manufacturer** – This is the name of the manufacturer of the device, shown in the device selection menu available by choosing **Project>Options>General Options>Target>Device** where the user selects a device for the project.
- **Name** – This is the name of the device, shown as a submenu of the manufacturer menu item in the device selection menu available by choosing **Project>Options>General Options>Target>Device**.

Code banking

This section can be left blank if the device does not support code banking:

- **Number of code banks** – The number of banks excluding what is usually called the *common bank* (which is usually located in the lower part of the code memory address range).

- **Bank select register** – The address of the SFR controlling which bank is active.
- **Bank select register mask** – A bit mask (hexadecimal notation) filtering out which bits in the bank select register are used for bank selection.
- **Bank start address** – The start address of the code bank area (not the *common bank*).
- **Bank size** – Size of a code bank, in bytes (hexadecimal notation).

Data pointer

General properties of the XDATA access mechanisms which are mandatory:

- **Size of DPTR** – 16 or 24 bits.
- **Number of DPTR** – A number between 1 and 8, depending on the device.
- **DPTR bits 16-23** – The address of the SFR containing the most significant bits of a 24-bit DPTR. This field can be left blank if the size of the DPTR is 16 bits.
- **Page register bits 8-15** – The address of the SFR containing the most significant address bits when using `MOVX A,@Ri` or `MOVX @Ri,A`, with 16-bit addressing (if 24-bit addressing is used it is the middle address bits). This field can be left blank if the device does not have external RAM.
- **Page register bits 16-23** – The address of the SFR containing the most significant address bits when using `MOVX A,@Ri` or `MOVX @Ri,A`, with 24-bit addressing. This field can be left blank if 16-bit addressing is used or if the device does not have external RAM.

Multiple DPTR

This section can be left blank if the device only supports one DPTR:

- **DPTR select register** – The address of the SFR controlling which DPTR is active.
- **DPTR select register mask** – A bit mask (hexadecimal notation) filtering out which bits in the DPTR select register are used for DPTR selection.
- **DPTR selectable by INC instruction** – Yes or no depending on whether the DPTR select register is constructed in a way which allows toggling the active DPTR by executing the `INC` *`DPTRSelectRegister`* instruction.
- **Separate address registers** – Yes or no depending on whether there are separate DPL and DPH SFRs for each DPTR.
- **DPTR*n* bits 0-7** – The address of the SFR representing DPL for the DPTR with number *n* (*n* > 0, 0 represents the default DPTR). This field can be left blank if there are no separate address registers.
- **DPTR*n* bits 8-15** – The address of the SFR representing DPH for the DPTR with number *n* (*n* > 0, 0 represents the default DPTR). This field can be left blank if there are no separate address registers.
- **DPTR*n* bits 16-23** – The address of the SFR representing DPX for the DPTR with number *n* (*n* > 0, 0 represents the default DPTR). This field can be left blank if there are no separate address registers.

Memory layout

This information is mandatory. It is used to set up a default linker configuration. The linker configuration needs to be further edited manually if more complex segment definitions are required.

- **External RAM start address** – Start address of the external RAM (XDATA) if any (hexadecimal notation) where the linker will locate application data. This field can be left blank if the device does not have external RAM.
- **External RAM size** – Size of the external RAM (XDATA) if any (hexadecimal notation). The size is the amount of memory (in bytes) available to the linker. This field can be left blank if the device does not have external RAM.
- **External RAM on-chip** – Yes or no depending on whether the RAM is integrated in the MCU or not. This field can be left blank if the device does not have external RAM.
- **Code memory start address** – Start address of the code memory where the linker will locate the application. For a device with banking capabilities, this is the start address of the *common bank*.
- **Code memory size** – Size of the code memory (hexadecimal notation). The size is the amount of memory (in bytes) available to the linker. This field can be left blank if the device has banking capabilities since it is assumed that the *common bank* area and the area of the other code banks are adjacent without any gap in the address space.

SFR definitions

This information is used to generate data structures for accessing SFRs from source code and support for viewing SFRs in the debugger. A row in the SFR definition table defines one SFR. Each column is described below.

- **Address** – The address of the SFR (hexadecimal notation without the prefix ‘0x’). It is possible to use the same address for another SFR definition. If the SFR name is different for each such definition this is typically used to take care of the scenario where different SFRs can be mapped to a specific address in runtime, depending on other circumstances such as the value of another SFR. If the SFR name is the same for each such definition this is typically used to take care of the scenario where one or more bits or bit fields have different semantics (see *Bit n name* below), depending on other circumstances such as the value of another SFR. The address is mandatory.
- **SFR name** – The name of the SFR as it is referenced in source code and viewed in the debugger. The same name must not be used for two SFRs with different addresses. The name is mandatory.
- **Bit *n* name** – Each bit in the SFR can have a name which makes it possible to access the individual bit in source code and debugger. Bit names are not required but should, if present, only contain alphanumerical characters and underscore. For example, an AD converter data register does not need any bit names, since the individual bits do not have any specific meaning. Bits which are reserved or not used should not have names. If two or more consecutive bits have the same name, they will form a bit-field which is accessed as a unit. It is possible to have more than one name for a bit if it has different semantics in different situations. This is done by defining the same SFR again (on a separate row in the table, with the same name, address and other data) but with different names for the bits which should have multiple names. In such secondary SFR definitions, the bits which do not have any alternative names should be left blank. Bit names must only be unique within the same SFR, different SFRs can reuse the same bit name.
- **Reset value** – The value of the SFR after a hardware reset (hexadecimal notation without the prefix ‘0x’). The reset value is mandatory.
- **Group name** – A name representing a group of related SFRs. This is used to make it more convenient to view SFRs in the debugger. This is optional and can be left blank but it will be harder to get an overview of the contents in the debugger window showing SFRs.

- **Tooltip** – A text string which is shown as a tooltip when hovering with the mouse pointer over a specific SFR in the debugger. Provides a more descriptive denotation of the SFR to the user. This field is optional and can be left blank.

Adjusting the memory layout

Adapting to a new memory layout requires modification of the linker configuration file. The linker configuration file contains options to the linker. These options are documented in the *Linker and Library Tools Reference Guide*. The linker configuration files are located in the directory `8051\config\devices\`*`manufacturer-name`* and are named `lnk51ew_`*`device-name`*`.xcl` or `lnk51ew_`*`device-name`*`_banked.xcl`. For information on how to use a customized linker configuration file in your project, read about linker options in the *IDE Project Management and Building Guide*.

External RAM (XDATA)

Modifying the size of the linkable XDATA memory space is done by editing the linker symbol definitions (`-D` option) which specify the XDATA start and end addresses. These are used when allocating segments for the linker (`-Z` and `-P` options).

- **Default XDATA** – The XDATA memory range is defined by the linker symbol definitions `_XDATA0_START` and `_XDATA0_END`. The address 0x0000 cannot be included in the linkable XDATA address range for reasons related to compiler design. So even if the real range starts at address 0x0000, the value of `_XDATA0_START` must be 0x0001.
- **Internal XDATA** – For convenience, on-chip XDATA memory (if applicable) is defined separately by the linker symbol definitions `_IXDATA0_START` and `_IXDATA0_END`. If the on-chip XDATA is the only XDATA memory, the default XDATA linker symbol definitions described above gets the same values (they are the same). As for default XDATA, the address 0x0000 must not be part of the linkable memory range.

Code memory

Modifying the size of the linkable code memory space is done by editing the linker symbol definitions (`-D` option) which specify the code memory start and end addresses. These are used when allocating segments for the linker (`-Z` and `-P` options).

- **Device without code banking capabilities** – The code memory range is defined by the linker symbol definitions `_CODE0_START` and `_CODE0_END`.
- **Device with code banking capabilities** – The banked code memory range is defined by the option **Bank start** and **Bank end** which are set on the **General Options>Code Bank** page in the **Options** dialog and are not defined in the linker configuration file. The memory range used when selecting the *Near* code model in the project options is defined by the linker symbol definition `_NEAR_CODE_START` (which is located in the linker configuration file) and *value-of-Bank-start* - 1. This range is usually the same as what is often called the *common bank*.

Adjusting SFR definitions

SFR (Special Function Registers) definitions are relevant in the debugger for the purpose of viewing and modifying their values in a debug session. From a source code point of view it is convenient to have a representation and definition of each SFR so that it can be accessed from source code. In the pre-defined

device support that comes with IAR Embedded Workbench, the debugger and source code SFR definitions have the same names so that it is easier to follow the value of an SFR.

SFR definitions for the debugger

There are two main ways of managing SFR definitions for the debugger, you can:

- Use the **SFR Setup** dialog box in the IAR Embedded Workbench IDE. The SFR definitions created this way are only valid for the current project. They are treated separately and do not become integrated into any existing device description file. For more information about the **SFR Setup** dialog box, see the *C-SPY Debugging Guide*.
- Customize the *device description file*. When a device description file has been modified, a project which refers to it must be reloaded for the changes to take effect. The default device description files which come with the IAR Embedded Workbench are located in the directory `8051\config\devices\`*`manufacturer-name`* and are named `io`*`device-name`*`.ddf`. For more information on how to use a customized device description file in your project, see the *C-SPY Debugging Guide*.

  Device description files are organized in several sections. Use the following information for defining SFRs for debugging scenarios using hardware:

  [Sfr] – Followed by one or more SFR definitions, each on a separate line. The SFR definition format is outlined below. Items enclosed in square brackets are optional. SFR bit or bit field definitions are optional but must have a definition of their own. Any bit or bit field definitions for an SFR must have the same name as the "parent" SFR and a bit name, separated by a dot (.).

  ```
  sfr = "name[.bit-name]", "zone", address, size, base=radix [,
  bitRange=range] [, tooltip="text"]
  ```

  `name` – The name of the SFR which must be unique among definitions without *`bit-name`*. The debugger is case sensitive when reading SFR names.

  `bit-name` – The name of a bit or bit field within an SFR. If this is used, *name* must exist in a previous definition with no *bit-name*. No two bit or bit field definitions for a given SFR can have the same *bit-name*. The debugger is case sensitive when reading bit or bit field names.

  `zone` – The name of the memory zone where the SFR resides. The name must be a valid memory zone in the debugger. For 8051 devices this is almost always `"SFR"` and in some cases where the SFR is mapped into XDATA, the zone should be `"XData"`. The debugger is case sensitive when reading zone names.

  `address` – The address of the SFR in hexadecimal format, with or without the prefix '0x'. Two or more SFR definitions, even without bit-name, can have the same address. This is typically used to take care of the scenario where different SFRs can be mapped to a specific address in runtime, depending on other circumstances such as the value of another SFR.

  `size` – The size of the SFR in bytes. For 8051 devices this is always 1.

  `radix` – The radix used for displaying the numeric value of the SFR. This value can be `2`, `8`, `10`, `16`, `float` or `double`.

  `range` – Only applicable if *`bit-name`* is used. This is either a single bit number (the least significant bit is numbered 0) representing one bit or a range on the form `i-j` (for example `0-2`) representing a bit-field.

  `text` – A text string which contains a user friendly description of the SFR, bit or bit field.

[SfrGroupInfo] – Followed by one or more SFR group definitions, each on a separate line. The SFR group definition format is outlined below. Items enclosed in square brackets are optional. SFR groups are used to organize the SFRs so that a better overview in the graphical user interface can be achieved. This section is optional.

```
group = "group-name", "sfr-name1" [, "sfr-name2", ...]
```

group-name – The name of a logical group of SFRs.

sfr-nameN – The name of an SFR which must be defined in the [Sfr] section. SFR bits and bit fields do not need to be included in a group since they are automatically attached to the relevant SFR in the user interface.

SFR definitions for source code

In IAR Embedded Workbench, all source code SFR definitions are collected in one include-file for a specific device (both for C/C++ and assembler). This is of course a matter of design and is not mandatory in any way. The default include-files which come with the IAR Embedded Workbench are located in the directory `8051\inc` and are named `io`*`device-name`*`.h`. SFR definitions for source code can be different depending on whether bit and/or bit fields should be represented. However, all SFR variables in C/C++ code should be declared using the keywords `__sfr` (or `__xdata` if the SFR is located in XDATA memory), `__no_init` and `volatile`. They should also be placed at an absolute address. Take a look in any of the include-files that come with the product.