Document Number: WG14 N706 (J11-069)

STANDARDIZATION OF BASIC I/O HARDWARE ADDRESSING.

Title: STANDARDIZATION OF BASIC I/O HARDWARE ADDRESSING.
Author: Jan Kristoffersen
Author Affiliation: RAMTEX International AS
Postal Address: Box 84, 2850 Naerum, Denmark
E-mail Address: jkristof@pip.dknet.dk
Telephone Number: +45 45505357
Fax Number: +45 45505390
Sponsor: DS
Date: May 22, 1997
Proposal Category:
   __ Editorial change/non-normative contribution
   __ Correction
   XX New feature
   __ Addition to obsolescent feature list
   __ Addition to Future Directions
   __ Other (please specify)
Area of Standard Affected:
   __ Environment
   __ Language
   __ Preprocessor
   XX Library
      __ Macro/typedef/tag name
      XX Function
      XX Header
   XX Other (please specify)  New Annex

---

Prior Art: Widely used
Target Audience: Device driver programmers, I/O chip manufactures

Related Documents (if any):

| | |
|---|---|
| WG14 N509/X3J11 95-110 | Introduction paper |
| WG14 N558/X3J11 95-110 | Proposal |
| WG14 N656/X3J11 | Revised Proposal |

-------------------------------

**Annex X (Draft 1.31)**
(informative)
**Basic I/O hardware addressing <iohw.h>**

# X.1 Introduction

[1] The header <iohw.h> declares several functions for making direct access to I/O registers in peripheral hardware. [1]

The purpose of these functions is to make device driver functions for peripheral I/O hardware more portable between different execution environments and implementations. [2]

## X.1.1 Conceptual model for I/O registers

[1] A given I/O register is always considered to have a fixed size in number of bits. This register size is independent of the execution environment and the implementation.
The I/O functions in <iohw.h> consider an I/O register to belong to one of the following categories: 1 bit, 8 bit, 16 bit, 32 bit, or 64 bit.

[2] The I/O functions in <iohw.h> deals with four basic operations on I/O registers: read, write, bit-wise AND, and bit-wise OR. [3]

[3] I/O registers, or individual bits in a register, are assumed to belong to one of the categories below. Each category places different restrictions on allowed operations.[4]

| Register category | Allowed operation |
|---|---|
| Read-Only | Read |
| Write-Only | Write |
| Read-and-Write [5] | Read, Write |
| Read-Modify-Write [6] | Read, Write, AND, OR |

The result of an operation not allowed by the register category shall be considered as undefined.

---

1 The term peripheral hardware includes, but is not limited to, I/O chips, I/O cells, and I/O circuitry.
A peripheral hardware device can be considered as a collection of individual I/O registers.

2 I/O driver portability is defined as follows:
a. If it is possible to reuse I/O hardware with another platform, then it should also be possible to reuse the hardware device drivers for that I/O hardware in the new platform, without any modifications to the driver source code.
b. If there exists two translators for the same platform, it should be possible to compile I/O hardware device driver source code with either translators, and get the same logical behaviors time of execution.

3 AND operations are used for clearing register bits. OR operations are used for setting register bits.

4 As to which access category an given I/O register belongs to depends of the physical characteristics of the register itself and its physical location in the execution environment.

5 Behaves like a read-only register and a write-only register on the same physical address.

6 Behaves like a memory cell.

## X.2 Access type parameter

[1] The I/O functions declared in <iohw.h> shall use a single parameter, the access type, as identification of an I/O register. The access type parameter represents or references a complete description of how a given register is connected in the given hardware platform with the given processor architecture. [7]

[2] Access type parameter definitions shall be included via the header <iohw.h>. [8]

### X.2.1 Access type parameter definitions

[1] The definition of access type parameters will be implementation defined. [9]

[2] A typical implementation will operate with the following five access type categories, one for each register size:

```
access1
access8
access16
access32
access64
```

[3] An I/O register definition shall contain the following elements:

1. A symbolic I/O register name

which represents a complete specification of the access method for the I/O register in the given hardware platform:

2. I/O register data size: (i.e., 1, 8, 16, 32, or 64 bits)
3. Address space used (if more than one address space)

4a. Physical address (if data size = register size)
    or
4b. Physical address, bit position. (if bit is in a register)

## X.3 I/O Functions

### X.3.1 I/O register read functions

### X.3.1.1 The iord1 function

---

7 That is, how the I/O register should be addressed.

8 This encapsulates the descriptions of processor and platform architecture differences.
The recommended practice is to let <iohw.h> contain I/O function definitions specific to the processor architecture, and let <iohw.h> include a separate header <iohw_ta.h> which contain access type definitions for all I/O registers in the given target platform.

9 An access type definition will typically depend on the given processor architecture, platform hardware, and translator.

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
bool iord1( access1 adr );
```

**Description**

[2] The iord1 function reads a value from an 1-bit register whose physical location is defined by the argument adr.

**Returns**

[3] The iord1 function returns the value read from the 1-bit register.

## X.3.1.2 The iord8 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least8_t iord8( access8 adr );
```

**Description**

[2] The iord8 function reads a value from an 8-bit register whose physical location is defined by the argument adr.

**Returns**

[3] The iord8 function returns the value read from the 8-bit register.

## X.3.1.3 The iord16 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least16_t iord16( access16 adr );
```

**Description**

[2] The iord16 function reads a value from a 16-bit register whose physical location is defined by the argument adr.

**Returns**

[3] The iord16 function returns the value read from the 16-bit register.

## X.3.1.4 The iord32 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least32_t iord32( access32 adr );
```

**Description**

[2] The iord32 function reads a value from a 32-bit register whose physical location is defined by the argument adr.

**Returns**

[3] The iord32 function returns the value read from the 32-bit register.

### X.3.1.5 The iord64 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least64_t iord64( access64 adr );
```

**Description**

[2] The iord64 function reads a value from a 64-bit register whose physical location is defined by the argument adr.

**Returns**

[3] The iord64 function returns the value read from the 64-bit register.

## X.3.2 I/O register write functions

### X.3.2.1 The iowr1 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void iowr1( access1 adr, bool value );
```

**Description**

[2] The iowr1 function writes the argument value to a 1-bit register whose physical location is defined by the argument adr.

### X.3.2.2 The iowr8 function

**Synopsis**

[1]
```
#include <inttypes.h>
#include <iohw.h>
void iowr8( access8 adr, uint_least8_t value );
```

**Description**

[2] The iowr8 function writes the argument value to an 8-bit register whose physical location is defined by the argument adr.

### X.3.2.3 The iowr16 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void iowr16( access16 adr, uint_least16_t value );
```

**Description**

[2] The iowr16 function writes the argument value to a 16-bit register whose physical location is defined by the argument adr.

### X.3.2.4 The iowr32 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void iowr32( access32 adr, uint_least32_t value );
```

**Description**

[2] The iowr32 function writes the argument value to a 32-bit register whose physical location is defined by the argument adr.

### X.3.2.5 The iowr64 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void iowr64( access64 adr, uint_least64_t value );
```

**Description**

[2] The iowr64 function writes the argument value to a 64-bit register whose physical location is defined by the argument adr.

### X.3.3 I/O register AND functions

### X.3.3.1 The ioand1 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioand1( access1 adr, bool value );
```

**Description**

[2] The ioand1 function performs a bit-wise AND of the argument value and a 1-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.3.2 The ioand8 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioand8( access8 adr, uint_least8_t value );
```

**Description**

[2] The ioand8 function performs a bit-wise AND of the argument value and an 8-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.3.3 The ioand16 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioand16( access16 adr, uint_least16_t value );
```

**Description**

[2] The ioand16 function performs a bit-wise AND of the argument value and a 16-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.3.4 The ioand32 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioand32( access32 adr, uint_least32_t value );
```

**Description**

[2] The ioand32 function performs a bit-wise AND of the argument value and a 32-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.3.5 The ioand64 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void ioand64( access64 adr, uint_least64_t value );
```

**Description**

[2] The ioand64 function performs a bit-wise AND of the argument value and a 64-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

## X.3.4 I/O register OR functions

### X.3.4.1 The ioor1 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void ioor1( access1 adr, bool value );
```

**Description**

[2] The ioor1 function performs a bit-wise OR of the argument value and a 1-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.4.2 The ioor8 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void ioor8( access8 adr, uint_least8_t value );
```

**Description**

[2] The ioor8 function performs a bit-wise OR of the argument value and an 8-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.4.3 The ioor16 function

**Synopsis**
[1]
```
#include <inttypes.h>
#include <iohw.h>
void ioor16( access16 adr, uint_least16_t value );
```

**Description**

[2] The ioor16 function performs a bit-wise OR of the argument value and a 16-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.4.4 The ioor32 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioor32( access32 adr, uint_least32_t value );
```

**Description**

[2] The ioor32 function performs a bit-wise OR of the argument value and a 32-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

### X.3.4.5 The ioor64 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void ioor64( access64 adr, uint_least64_t value );
```

**Description**

[2] The ioor64 function performs a bit-wise OR of the argument value and a 64-bit register whose physical location is defined by the argument adr. The result of the operation is stored in the register.

## X.3.5 I/O buffer read functions

### X.3.5.1 The iordbuf8 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least8_t iordbuf8( access8 adr, unsigned int index );
```

**Description**

[2] The iordbuf8 function reads the value of the [index] element in a hardware buffer arranged as a one-dimensional array of 8-bit registers. The physical location of element [0] is defined by the argument adr.

**Returns**

[3] The iordbuf8 function returns the value read from the 8-bit register.

### X.3.5.2 The iordbuf16 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least16_t iordbuf16( access16 adr, unsigned int index );
```

**Description**

[2] The iordbuf16 function reads the value of the [index] element in a hardware buffer arranged as a one-dimensional array of 16-bit registers. The physical location of element [0] is defined by the argument adr.

**Returns**

[3] The iordbuf16 function returns the value read from the 16-bit register.

## X.3.5.3 The iordbuf32 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least32_t iordbuf32( access32 adr, unsigned int index );
```

**Description**

[2] The iordbuf32 function reads the value of the [index] element in a hardware buffer arranged as a one-dimensional array of 32-bit registers. The physical location of element [0] is defined by the argument adr.

**Returns**

[3] The iordbuf32 function returns the value read from the 32-bit register.

## X.3.5.4 The iordbuf64 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
uint_least64_t iordbuf64( access64 adr, unsigned int index );
```

**Description**

[2] The iordbuf64 function reads the value of the [index] element in a hardware buffer arranged as a one-dimensional array of 64-bit registers. The physical location of element [0] is defined by the argument adr.

**Returns**

[3] The iordbuf64 function returns the value read from the 64-bit register.

## X.3.6 I/O buffer write functions

### X.3.6.1 The iowrbuf8 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void iowr8( access8 adr, unsigned int index, uint_least8_t value );
```

**Description**

[2] The iowrbuf8 function writes value to the [index] element in a hardware buffer arranged as a one-dimensional array of 8-bit registers. The physical location of element [0] is defined by the argument adr.

### X.3.6.2 The iowrbuf16 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void iowr16( access16 adr, unsigned int index, uint_least16_t value );
```

**Description**

[2] The iowrbuf16 function writes value to the [index] element in a hardware buffer arranged as a one-dimensional array of 16-bit registers. The physical location of element [0] is defined by the argument adr.

### X.3.6.3 The iowrbuf32 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void iowr32( access32 adr, unsigned int index, uint_least32_t value );
```

**Description**

[2] The iowrbuf32 function writes value to the [index] element in a hardware buffer arranged as a one-dimensional array of 32-bit registers. The physical location of element [0] is defined by the argument adr.

### X.3.6.4 The iowrbuf64 function

**Synopsis**

[1]

```
#include <inttypes.h>
#include <iohw.h>
void iowr64( access64 adr, unsigned int index, uint_least64_t value );
```