## Intrinsic functions

The following is a list of all intrinsic functions. Intrinsic function do not belong to the standard C language but the compiler may support intrinsics for a specific processor.

| | **ARM** | **Micro Blaze** | **Nios II** | **Power PC** | **TSK51x/ TSK52x** | **TSK80x** | **TSK3000** | |
|---|---|---|---|---|---|---|---|---|
| __alloc | x | x | x | x | x | x | x | Allocate memory |
| __break | | | | | | | x | Insert break instruction |
| __dotdotdot__ | | x | x | | x | x | | Variable argument '...' operator |
| __free | x | x | x | x | x | x | x | Deallocate memory |
| __getbit | | | | | x | | | Get the value of a bit |
| __putbit | | | | | x | | | Set the value of a bit |
| __get_return_address | x | x | x | x | x | | x | Function return address (when profiling) |
| __getapsr | x | | | | | | | Get APSR status register |
| __setapsr | x | | | | | | | Set APSR status register |
| __getcpsr | x | | | | | | | Get CPSR status register |
| __getipsr | x | | | | | | | Get IPSR status register |
| __setcpsr | x | | | | | | | Set CPSR status register |
| __getspsr | x | | | | | | | Get SPSR status register |
| __setspsr | x | | | | | | | Set SPSR status register |
| __cgetfsl | | x | | | | | | Read control words from fast simplex link |
| __cputfsl | | x | | | | | | Write control words to fast simplex link |
| __getfsl | | x | | | | | | Read data words from fast simplex link |
| __putfsl | | x | | | | | | Write data words to fast simplex link |
| __getfsr | | x | | | | | | Get FSR register |
| __putfsr | | x | | | | | | Set FSR register |
| __getmsr | | x | | | | | | Get MSR register |
| __putmsr | | x | | | | | | Set MSR register |
| __msrclr | | x | | | | | | Clear bits in MSR register |
| __msrset | | x | | | | | | Set bits in MSR register |
| __getpc | | x | | | | | | Get value of program counter PC |
| __mfspr | | | | x | | | | Get special function register |
| __mtspr | | | | x | | | | Set special function register |
| __mfctr | | | | x | | | | Get special function register CTR |
| __mtctr | | | | x | | | | Set special function register CTR |
| __mflr | | | | x | | | | Get special function register LR |
| __mtlr | | | | x | | | | Set special function register LR |
| __mfmsr | | | | x | | | | Get special function register MSR |
| __mtmsr | | | | x | | | | Set special function register MSR |
| __mfxer | | | | x | | | | Get special function register XER |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| __mtxer | | | x | | | | Set special function register XER |
| __getsp | | | | | x | | Get stack pointer (SP) |
| __setsp | | | | | x | | Set stack pointer (SP) |
| __mfc0 | | | | | | x | Get value from SPR of coprocessor 0 |
| __mtc0 | | | | | | x | Set value to SPR of coprocessor 0 |
| __nop | x | x | | x | | x | Insert NOP instruction |
| __rol | | | | x | | | Rotate left |
| __ror | | | | x | | | Rotate right |
| __svc | x | | | | | | Generate software interrupt. |
| __testclear | | | | x | | | Read and clear semaphore |
| __vsp__ | | | | x | | | Virtual Stack Pointer in use |

## Intrinsic function: __alloc

### Syntax

```
void * volatile __alloc( __size_t size );
```

Allocate memory. Same as library function malloc().

**Returns:** a pointer to space in external memory of size bytes length. NULL if there is not enough space left.

## Intrinsic function: __break

### Syntax

```
volatile int __break(int val);
```

Generates the assembly break instruction. *Val* is a 20-bit value which will be encoded in the code field of the break instruction..

**Returns:** nothing.

## Intrinsic function: __cgetfsl (MicroBlaze)

### Syntax

```
_Bool volatile __cgetfsl( unsigned char channel,
                          unsigned int * ctrl, _Bool wait );
```

Read control words from a specified fast simplex link (fsl) channel.

**Returns:** True if valid data was read from the specified channel, otherwise False.

## Intrinsic function: __cputfsl (MicroBlaze)

### Syntax

```
_Bool volatile __cputfsl( unsigned char channel,
                          unsigned int * ctrl, _Bool wait );
```

Write control words to a specified fast simplex link (fsl) channel.

**Returns:** True if valid data was read from the specified channel, otherwise False.

## Intrinsic function: __dotdotdot__

### Syntax

```
char * __dotdotdot__( void );
```

Variable argument '...' operator. Used in library function va_start().

**Returns:** the stack offset to the variable argument list.

## Intrinsic function: __dotdotdot__ (Nios II)

### Syntax

```
void * __dotdotdot__( void );
```

Variable argument '...' operator. Used in library function va_start().

**Returns:** the stack offset to the variable argument list.

## Intrinsic function: __free

### Syntax

```
void volatile __free( void *p );
```

Deallocates the memory pointed to by p. p must point to memory earlier allocated by a call to __alloc(). Same as library function free().

**Returns:** nothing.

## Intrinsic function: __get_return_address

### Syntax

```
__codeptr volatile __get_return_address( void );
```

Used by the compiler for profiling when you compile with the **-p (--profile)** option.

**Returns:** return address of a function.

## Intrinsic function: __getapsr (ARM)

### Syntax

```
unsigned int volatile __getapsr( void );
```

**Note:** This intrinsic is only available for ARMv6-M and ARMv7-M (M-profile architectures).

Get the value of the APSR status register.

**Returns:** the value of the status register APSR.

## Intrinsic function: __getbit (TSK51x/TSK52x)

### Syntax

```
__bit __getbit( bitaddressable, bitoffset );
```

Get the value of a bit. *`bitoffset`* must be an integral constant expression.

**Returns:** the bit at *`bitoffset`* (range 0-7 for a `char`, 0-15 for an int or 0-31 for a long) of the *`bitaddressable`* operand for use in bit expressions.

### Example

```
__bdata unsigned char byte;
int i;

if ( __getbit( byte, 3 ) )
    i = 1;
```

## Intrinsic function: __getcpsr (ARM)

### Syntax

```
unsigned int volatile __getcpsr( void );
```

Get the value of the CPSR status register.

**Returns:** the value of the status register CPSR.

## Intrinsic function: __getfsl (MicroBlaze)

### Syntax

```
_Bool volatile __getfsl( unsigned char channel,
                         unsigned int * data, _Bool wait );
```

Read data words from a specified fast simplex link (fsl) channel. *`Channel`* must be a constant value in the range 0..7. The read data is stored in `*data`. With the boolean *`wait`* you can choose whether or not to wait for information: True: wait for information, False: do not wait for information (the channel may not provide data).

**Returns:** True if valid data was read from the specified channel, otherwise False.

## Intrinsic function: __getfsr (MicroBlaze)

### Syntax

```
unsigned int volatile __getfsr( void );
```

Get the value of the floating-point state register FSR.

**Returns:** the value of the floating-point state register FSR.

## Intrinsic function: __getipsr (ARM)

### Syntax

```
unsigned int volatile __getipsr( void );
```

**Note:** This intrinsic is only available for ARMv6-M and ARMv7-M (M-profile architectures).

Get the value of the IPSR status register.

**Returns:** the value of the status register IPSR.

## Intrinsic function: __getmsr (MicroBlaze)

### Syntax

```
unsigned int volatile __getmsr( void );
```

Get the value of the machine state register MSR.

**Returns:** the value of the machine state register MSR.

## Intrinsic function: __getpc (MicroBlaze)

### Syntax

```
unsigned int volatile __getpc( void );
```

Get the value of the program counter PC.

**Returns:** the value of the program counter.

## Intrinsic function: __getsp (TSK80x)

### Syntax

```
unsigned int volatile __getsp( void );
```

Get the value of the stack pointer SP.

**Returns:** the value of the stack pointer.

## Intrinsic function: __getspsr (ARM)

### Syntax

```
unsigned int volatile __getspsr( void );
```

Get the value of the SPSR status register.

**Returns:** the value of the status register SPSR.

### Example

```
#define SR_F 0x00000040
#define SR_I 0x00000080

i = __setspsr (0, SR_F | SR_I);
if (i & (SR_F | SR_I))
{
  exit (6);    /* Interrupt flags not correct */
  }

if (__getspsr () & (SR_F | SR_I))
{
  exit (7);    /* Interrupt flags not correct */
  }
```

## Intrinsic function: __mfc0 (TSK3000)

### Syntax

```
volatile int __mfc0(int spr);
```

Get the value from special function register *spr* of coprocessor 0.

**Returns:** the value of the SPR register of coprocessor 0.

## Intrinsic function: __mfctr (PowerPC)

### Syntax

```
volatile int __mfctr(void);
```

Get the value from special function register CTR. (This equivalent to `__mfspr(0x009)` )

**Returns:** the value of the CTR register.

## Intrinsic function: __mflr (PowerPC)

### Syntax

```
volatile int __mflr(void);
```

Get the value from special function register LR. (This equivalent to `__mfspr(0x008)` )

**Returns:** the value of the LR register.

## Intrinsic function: __mfmsr (PowerPC)

### Syntax

```
volatile int __mfmsr(void);
```

Get the value from special function register MSR.

**Returns:** the value of the MSR register.

## Intrinsic function: __mfspr (PowerPC)

### Syntax

```
volatile int __mfspr(int spr);
```

Get the value from a special function register. *`spr`* is the number of the special purpose register and can be specified either as a decimal number or as a hexadecimal number.

**Returns:** the value of the specified special purpose register.

## Intrinsic function: __mfxer (PowerPC)

### Syntax

```
volatile int __mfxer(void);
```

Get the value from special function register XER. (This equivalent to `__mfspr(0x001)` )

**Returns:** the value of the XER register.

## Intrinsic function: __msrclr (MicroBlaze)

### Syntax

```
unsigned int __msrclr( unsigned int value );
```

Clear a number of bits in the machine state register MSR. *`Value`* should be a 14 bit mask. If you specify a value larger than $2^{14}$, the instruction is ignored and the compiler will use the `getmsr` and `putmsr` instructions instead.

**Returns:** the value of the MSR register before bits were cleared.

## Intrinsic function: __msrset (MicroBlaze)

### Syntax

```
unsigned int __msrset( unsigned int value );
```

Set a number of bits in the machine state register MSR. *`Value`* should be a 14 bit mask. If you specify a value larger than $2^{14}$, the instruction is ignored and the compiler will use the `getmsr` and `putmsr` instructions instead.

**Returns:** the value of the MSR register before bits were set.

## Intrinsic function: __mtc0 (TSK3000)

### Syntax

```
volatile void __mtc0(int val, int spr);
```

Put a value *`val`* into special purpose register *`spr`* of coprocessor 0.

**Returns:** nothing.

## Intrinsic function: __mtctr (PowerPC)

### Syntax

```
volatile void __mtctr(int val);
```

Put a value *val* into special function register CTR. (This equivalent to `__mtspr(0x009,val)` )

**Returns:** nothing.

## Intrinsic function: __mtlr (PowerPC)

### Syntax

```
volatile void __mtlr(int val);
```

Put a value *val* into special function register LR. (This equivalent to `__mtspr(0x008,val)` )

**Returns:** nothing.

## Intrinsic function: __mtmsr (PowerPC)

### Syntax

```
volatile void __mtmsr(int val);
```

Put a value *val* into special function register MSR.

**Returns:** nothing.

## Intrinsic function: __mtspr (PowerPC)

### Syntax

```
volatile void __mtspr(int spr, int val);
```

Put a value into a special function register. *spr* is the number of the special purpose register and can be specified either as a decimal number or as a hexadecimal number. *val* is the value to put into the specified register.

**Returns:** nothing.

## Intrinsic function: __mtxer (PowerPC)

### Syntax

```
volatile void __mtxer(int val);
```

Put a value *val* into special function register XER. (This equivalent to `__mtspr(0x001,val)` )

**Returns:** nothing.

## Intrinsic function: __nop

### Syntax

```
void __nop( void );
```

Generate NOP instructions.

**Returns:** nothing.

### Example

```
__nop();          /* generate NOP instruction */
```

## Intrinsic function: __putbit (TSK51x/TSK52x)

### Syntax

```
void __putbit( __bit value, bitaddressable, bitoffset );
```

Assign a *value* to the bit at *bitoffset* (range 0-7 for a char , 0-15 for an int or 0-31 for a long) of the *bitaddressable* operand. *bitoffset* must be an integral constant expression.

**Returns:** nothing.

### Example

```
__bdata unsigned int word;

__putbit( 1, word, 11 );
__putbit( 0, word, 10 );
```

## Intrinsic function: __putfsl (MicroBlaze)

### Syntax

```
_Bool volatile __putfsl( unsigned char channel,
                         unsigned int * data, _Bool wait );
```

Write data words to a specified fast simplex link (fsl) channel. *Channel* must be a constant value in the range 0..7. The data to write must be stored in `*data`. With the boolean *wait* you can choose whether or not to wait for information: True: wait for information, False: do not wait for information (the channel may not accept data).

**Returns:** True if valid data was written to the specified channel, otherwise False.

## Intrinsic function: __putfsr (MicroBlaze)

### Syntax

```
void volatile __putfsr( unsigned int value );
```

Set the *value* of the floating-point state register FSR to value.

**Returns:** nothing.

## Intrinsic function: __putmsr (MicroBlaze)

### Syntax

```
void volatile __putmsr( unsigned int value );
```

Set the value of the machine state register MSR to `value`.

**Returns:** nothing.

## Intrinsic function: __rol (TSK51x/TSK52x)

### Syntax

```
unsigned char __rol( unsigned char operand, unsigned char count );
```

Use the RL instruction to rotate *`operand`* left *`count`* times.

**Returns:** rotated value.

### Example

```
unsigned char c;
int i;

/* rotate left, using int variable */
c = __rol( c, i );
```

## Intrinsic function: __ror (TSK51x/TSK52x)

### Syntax

```
unsigned char __ror( unsigned char operand, unsigned char count );
```

Use the RR instruction to rotate *`operand`* right *`count`* times.

**Returns:** rotated value.

### Example

```
unsigned char c;
int i;

/* rotate right, using constant */
c = __ror( c, 2 );
c = __ror( c, 3 );
c = __ror( c, 7 );
```

## Intrinsic function: __setapsr (ARM)

### Syntax

```
unsigned int volatile __getapsr( void );
```

**Note:** This intrinsic is only available for ARMv6-M and ARMv7-M (M-profile architectures).

Set or clear bits in the APSR status register.

**Returns:** the new value of the APSR status register.

## Intrinsic function: __setcpsr (ARM)

### Syntax

```
unsigned int volatile __setcpsr( int set, int clear);
```

Set or clear bits in the CPSR status register.

**Returns:** the new value of the CPSR status register.

## Intrinsic function: __setsp (TSK80x)

### Syntax

```
void volatile __setsp( unsigned int value );
```

Set the value of the stack pointer SP to *`value`*.

**Returns:** nothing.

## Intrinsic function: __setspsr (ARM)

### Syntax

```
unsigned int volatile __setspsr( int set, int clear);
```

Set or clear bits in the SPSR status register.

**Returns:** the new value of the SPSR status register.

### Example

```
#define SR_F 0x00000040
#define SR_I 0x00000080

i = __setspsr (0, SR_F | SR_I);
if (i & (SR_F | SR_I))
{
  exit (6);    /* Interrupt flags not correct */
  }

if (__getspsr () & (SR_F | SR_I))
{
  exit (7);    /* Interrupt flags not correct */
  }
```

## Intrinsic function: __svc (ARM)

### Syntax

```
void volatile __svc(int number);
```

Generates a supervisor call (software interrupt). *`Number`* must be a constant value.

**Returns:** nothing.

## Intrinsic function: __testclear (TSK51x/TSK52x)

### Syntax

```
__bit volatile __testclear( __bit *semaphore );
```

Read and clear `semaphore` using the JBC instruction.
**Returns:** 0 if `semaphore` was not cleared by the JBC instruction, 1 otherwise.

### Example

```
__bit b;
unsigned char c;

if ( __testclear( &b ) )              /* JBC instruction */
  c=1;
```

## Intrinsic function: __vsp__ (TSK51x/TSK52x)

### Syntax

```
__bit __vsp__( void );
```

Virtual stack pointer used. Used in library function va_arg().

**Returns:** 1 if the virtual stack pointer is used, 0 otherwise.