UnoLib documentation

defs.pas

version 10/26/2025

The defs.pas module contains low-level constants for registers, routines for managing bits, and access to I/O ports. Routines were originally preprocessor functions (macros) in the Arduino core library, but because FPC doesn't support such routines, they were translated as regular procedures or functions. Consequently, the binary code produced by FPC is slightly larger than the Arduino code.

Routines
`function` **`ByteToHex`**`(val: UInt8): shortstring;` Returns hexadecimal representation of *val* as shortstring type. Note: function probably to move to stringutils.pas
`function` **`BitIsSet`**`(const aSfrp: PUInt8; const aBit: UInt8): boolean;` Returns *true* if specified bit of byte value is set. *Parameters* *aSfrp*: pointer to a byte value (PUInt8). *aBit*: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.
`function` **`BitIsClear`**`(const aSfrp: PUInt8; const aBit: UInt8): boolean;` Returns *true* if specified bit of byte value is clear (not set). *Parameters* *aSfrp*: pointer to a byte value (PUInt8). *aBit*: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.
`function` **`BitRead`**`(const value, bit: UInt8): UInt8;` Returns a bit value (0 or 1) of a variable on a specific position. *Parameters* *value*: byte value (UInt8). *bit*: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.
`procedure` **`BitSet`**`(var value: UInt8; const bit: UInt8);` Sets (assigns to 1) the bit state on the *bit* position of the *value*.

Parameters
value: byte value (UInt8).
bit: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.

```
procedure BitWrite(var value: UInt8; const bit, bitvalue: UInt8);
```

Sets a specific value (0 or 1) to the bit state at the specified bit position of the byte value.

Parameters
value: byte value (UInt8).
bit: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.
bitvalue: value of the bit. Valid range is 0 to 1.

```
function _BV(const aBit: UInt8): UInt8;
```

Returns a byte with a specific bit set (assigned to 1).

Parameters
aBit: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.

```
procedure Cbi(const aSfrp: PUInt8; const aBit: UInt8);
```

Clears (sets to 0) a specific bit position of the byte value.

Parameters

aSfrp: pointer to a byte value (PUInt8).
aBit: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.

```
procedure Sbi(const aSfrp: PUInt8; const aBit: UInt8);
```

Sets (assigns to 1) a specific bit position of the byte value.

Parameters

aSfrp: pointer to a byte value (PUInt8).
aBit: index of a bit, starting at 0 for the least-significant (rightmost) bit. Valid range is 0 to 7.

```
function PortModeRegister(const aPort: UInt8): PUInt8;
```

Returns address of Port Mode Register (Data Direction Register in AVR architecture).

Parameters

aPort: constant defining AVR I/O port. Valid values are: PB, PC, PD.

```
function PortInputRegister(const aPort: UInt8): PUint8;
```

Returns address of Port Input Register.

Parameters

aPort: constant defining AVR I/O port. Valid values are: PB, PC, PD.
`function` **`PortOutputRegister`**`(const aPort: UInt8): PUint8;` Returns address of Port Output Register. *Parameters* *aPort*: constant defining AVR I/O port. Valid values are: PB, PC, PD.