Register header quick reference

Introduction

All names used in the register macros exactly match the reference manual. The only exceptions are cases where the names in the manual are not valid C language identifiers, or there are duplicate names within a given scope.

Some modules in the chips have multiple instances. For example, there are five eCSPI modules in the i.MX6DQ. The macros used to access registers of modules with more than one instance differ slightly from macros for modules with a single instance. Most macros take an additional instance number argument as the first argument. Instances are numbered starting at 1.

Register Macros

Each register has a struct definition (actually a union) containing all of the bitfields. In addition, there is an address macro and macros to read and write the register. Not all macros will be available for registers that are read-only or write-only.

Single-instance modules

Format	Purpose	Example
`hw_<module>_<register>_t`	register struct	`hw_gpmi_ctrl0_t`
`HW_<module>_<register>_ADDR`	register address	`HW_GPMI_CTRL0_ADDR`
`HW_<module>_<register>`	access register struct	`HW_GPMI_CTRL0`
`HW_<module>_<register>_RD()`	read register	`HW_GPMI_CTRL0_RD()`
`HW_<module>_<register>_WR(v)`	write register	`HW_GPMI_CTRL0_WR(0xc0000000)`
`HW_<module>_<register>_SET(v)`	set register bits	`HW_GPMI_CTRL0_SET(0x1)`
`HW_<module>_<register>_TOG(v)`	clear register bits	`HW_GPMI_CTRL0_CLR(0x1)`
`HW_<module>_<register>_CLR(v)`	toggle register bits	`HW_GPMI_CTRL0_TOG(0x1)`

Multi-instance modules

Macros for multi-instance modules take the instance number as an additional first argument.

Format	Purpose	Example
`hw_<module>_<register>_t`	register struct	`hw_ecspi_conreg_t`
`HW_<module>_<register>_ADDR(x)`	register address	`HW_ECSPI_CONREG_ADDR(1)`
`HW_<module>_<register>(x)`	access register struct	`HW_ECSPI_CONREG(1)`
`HW_<module>_<register>_RD(x)`	read register	`HW_ECSPI_CONREG_RD(1)`
`HW_<module>_<register>_WR(x, v)`	write register	`HW_ECSPI_CONREG_WR(1, 0x1000)`
`HW_<module>_<register>_SET(x, v)`	set register bits	`HW_ECSPI_CONREG_SET(1, 0x1000)`
`HW_<module>_<register>_TOG(x, v)`	clear register bits	`HW_ECSPI_CONREG_CLR(1, 0x1000)`
`HW_<module>_<register>_CLR(x, v)`	toggle register bits	`HW_ECSPI_CONREG_TOG(1, 0x1000)`

Register struct

The register struct allows easy access to the bitfields of a register, as well as the register value as a whole.

```
// Integer value of the register
HW_GPMI_CTRL0.U       // single-instance
HW_ECSPI_CONREG(1).U  // multi-instance

// Bitfield access
HW_GPMI_CTRL0.B.CLKGATE               // single-instance
HW_ECSPI_CONREG(1).B.CHANNEL_SELECT   // multi-instance
```

Bitfield Macros

Single-instance modules

Format	Purpose	Example
BP_*<module>*_*<register>*_*<field>*	bit position	BP_ECSPI_CONREG_CHANNEL_SELECT
BM_*<module>*_*<register>*_*<field>*	bit mask, pre-shifted	BM_ECSPI_CONREG_CHANNEL_SELECT
BF_*<module>*_*<register>*_*<field>*(v)	shift and mask bitfield value	BF_ECSPI_CONREG_CHANNEL_SELECT(2)
BG_*<module>*_*<register>*_*<field>*(r)	get bitfield value from register value	BG_ECSPI_CONREG_CHANNEL_SELECT(value)
BW_*<module>*_*<register>*_*<field>*(v)	write bitfield using SCT or RMW	BW_VDOA_VDOAC_BNDM(value)
BV_*<module>*_*<register>*_*<field>*__*<value>*	bitfield value constant	BV_VDOA_VDOAC_BNDM__BAND_HEIGHT_8

Multi-instance modules

Only the **BW_** macro differs for multi-instance modules.

Format	Purpose	Example
BW_*<module>*_*<register>*_*<field>*(x, v)	write bitfield using SCT or RMW	BW_VDOA_VDOAC_BNDM(1, value)