Regulator Machine Driver Interface

```
===================================
```

The regulator machine driver interface is intended for board/machine specific initialisation code to configure the regulator subsystem.

Consider the following machine :-

```
  Regulator-1 -+-> Regulator-2 --> [Consumer A @ 1.8 - 2.0V]
               |
               +-> [Consumer B @ 3.3V]
```

The drivers for consumers A & B must be mapped to the correct regulator in order to control their power supply. This mapping can be achieved in machine initialisation code by creating a struct regulator_consumer_supply for each regulator.

```
struct regulator_consumer_supply {
        struct device *dev;     /* consumer */
        const char *supply;     /* consumer supply - e.g. "vcc" */
};
```

e.g. for the machine above

```
static struct regulator_consumer_supply regulator1_consumers[] = {
{
        .dev    = &platform_consumerB_device.dev,
        .supply = "Vcc",
},};

static struct regulator_consumer_supply regulator2_consumers[] = {
{
        .dev    = &platform_consumerA_device.dev,
        .supply = "Vcc",
},};
```

This maps Regulator-1 to the 'Vcc' supply for Consumer B and maps Regulator-2 to the 'Vcc' supply for Consumer A.

Constraints can now be registered by defining a struct regulator_init_data for each regulator power domain. This structure also maps the consumers to their supply regulator :-

```
static struct regulator_init_data regulator1_data = {
        .constraints = {
                .min_uV = 3300000,
                .max_uV = 3300000,
                .valid_modes_mask = REGULATOR_MODE_NORMAL,
        },
        .num_consumer_supplies = ARRAY_SIZE(regulator1_consumers),
        .consumer_supplies = regulator1_consumers,
};
```

Regulator-1 supplies power to Regulator-2. This relationship must be registered with the core so that Regulator-1 is also enabled when Consumer A enables its supply (Regulator-2). The supply regulator is set by the supply_regulator_dev field below:-

```
static struct regulator_init_data regulator2_data = {
	.supply_regulator_dev = &platform_regulator1_device.dev,
	.constraints = {
		.min_uV = 1800000,
		.max_uV = 2000000,
		.valid_ops_mask = REGULATOR_CHANGE_VOLTAGE,
		.valid_modes_mask = REGULATOR_MODE_NORMAL,
	},
	.num_consumer_supplies = ARRAY_SIZE(regulator2_consumers),
	.consumer_supplies = regulator2_consumers,
};
```

Finally the regulator devices must be registered in the usual manner.

```
static struct platform_device regulator_devices[] = {
{
	.name = "regulator",
	.id = DCDC_1,
	.dev = {
		.platform_data = &regulator1_data,
	},
},
{
	.name = "regulator",
	.id = DCDC_2,
	.dev = {
		.platform_data = &regulator2_data,
	},
},
};
/* register regulator 1 device */
platform_device_register(&regulator_devices[0]);

/* register regulator 2 device */
platform_device_register(&regulator_devices[1]);
```