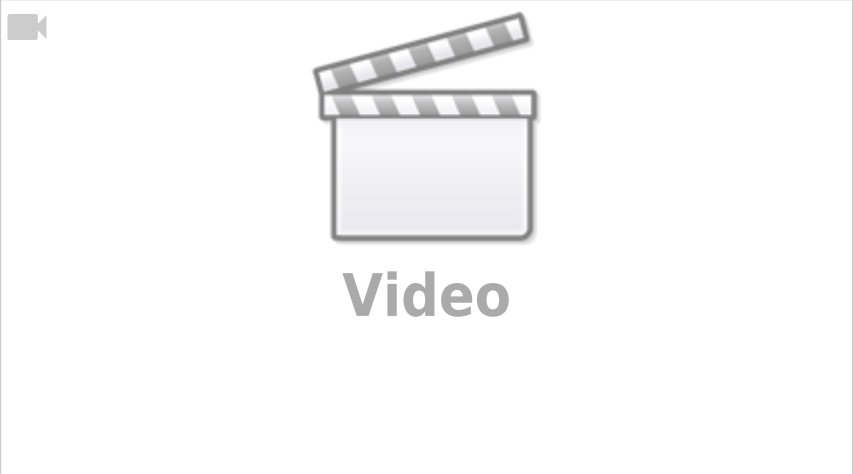
Global Variables - Hardware I/O Ports (#7180-7187)

This section is a subtopic of Global Variables manual.

A video on the topic is available here:



The global variables 7180-7183 store the input values in 32 bit increments. Therefore, each register contains 32 bits corresponding to 32 input ports. In this way, Input0 is 31-0, Input1 is 63-32, etc.

Note the way the bits are arranged:

Variable	Name	Comment
7180	GVAR_HW_INPUTS0	bit 31...bit0 - note that the bits are laid out from 31 to 0 “left to right” rather than the other way around. This is important for the left shift (vs right shift) later in the code examples.
7181	GVAR_HW_INPUTS1	bit 63...bit32
7182	GVAR_HW_INPUTS2	bit 95...bit64
7183	GVAR_HW_INPUTS3	bit 127...bit96

Similarly, global variables 7184-7187 store the information on the state of the output ports, with each variable containing the state of 32 ports:

Variable	Name	Comment
7184	GVAR_HW_OUTPUTS0	bit 31...bit0 - note that the bits are laid out from 31 to 0 “left to right” rather than the other way around. This is important for the left shift (vs right shift) later in the code examples.
7185	GVAR_HW_OUTPUTS1	bit 63...bit32
7186	GVAR_HW_OUTPUTS2	bit 63...bit32
7187	GVAR_HW_OUTPUTS3	bit 127...bit96

An example of code using the global variable 7184 to access the state of the output:

```
# define TEST_OUTPUT 12
```

```
main()
{
port_state_0=gvarget(7184);

pin_state=port_state_0&(1<<TEST_OUTPUT);
if (pin_state==0)
{
   portset(TEST_OUTPUT);
  }else
  {
    portclr(TEST_OUTPUT);
  };

exit(99);
};
```

Note: we are using output 12 in this example. Any output can be used.

The code essentially consists of the following:

- Store the current value of global variable 7184 (states of output ports 31-0) in variable `port_state_0`
- Create a variable `pin_state` which will be equal to the `port_state_0` being "compared" to a value of 1 being "left shifted" by the the value stored in TEST_OUTPUT (in our case, it's left shifted by 12 places).
    - *The Binary AND (&) Operator copies a bit to the result if it exists in both operands. For instance, 0001&0010 would return 0, while 0101&0100 would return 100.*
    - *You can read up on the left shift operation here or consult the video above for more information.*
- The variable `pin_state` is then checked. If it is equal to zero, meaning that the current state of the specific bit within gvariable 7184 is equal to zero (in our case, the 12th bit from the right in the register), then the output is toggled ON. If the value of `pin_state` is not equal to zero, meaning that the 12th bit from the right in the 7184 register was equal to 1, then the output is toggled OFF.
- This allows us to toggle the output ON or OFF depending on its current state.

Another code example utilizing a similar idea is available below:

```
#define OUTPUT_LASET_SHUTTER 15

main()
{

a=gvarget(7184);
a=a&(1<<OUTPUT_LASER_SHUTTER);

if (a==0)
{
portset(OUTPUT_LASER_SHUTTER);
}else
{
```

```
portclr(OUTPUT_LASER_SHUTTER);
};

exit(99);
};
```