

Home

EditNew Page

alinagithub edited this page a minute ago · 12 revisions

Welcome to the LibShift595Arduino wiki!

This library is intended to simply controlling cascaded 595 shift registers.

Arduino sample code:

```
#include <LibShift595Arduino.h>

#define NumberOfRegisters 2
#define LatchPin          4
#define ClockPin           3
#define DataPin            2

// Instantiate the shift register object
Shift595Arduino MyShiftReg(DataPin, ClockPin, LatchPin, NumberOfRegisters);

void setup()
{
    MyShiftReg.Setup();
}

void loop()
{
    myShiftRegister.auto_update = false;
    myShiftRegister.OffAllRegisters();
    myShiftRegister.OnSingleAllRegisters(7);
    myShiftRegister.OnSingleAllRegisters(12);
    myShiftRegister.Update();
    delay(1000);
}
```


▼ Pages 1


Home


+ Add a custom sidebar


Clone this wiki locally


https://github.com/alinagith



 Clone in Desktop




















Help:

Shift register allocation and initialization::

Shift595Arduino(int datapin, int clockpin, int latchpin, int nbregisters=1);

```
/*
Allocation function:
    - datapin: specifies the Arduino pin connected to the 595 data input
    - clockpin: specifies the Arduino pin connected to the 595 clock input
    - latchpin: specifies the Arduino pin connected to the 595 latch output
    - nbregisters: specifies the number of cascaded registers. Default is 1.

ex: Shift595Arduino myShiftRegister(2, 3, 4, 2);
*/
```

boolean Setup();

```
/*
Setup function:
  Inits the Arduino pins.
  This function must be called in the setup()
  function of the Arduino program.

ex: myShiftRegister.Setup();
*/
```



Shift register actions::



boolean OffRegister(int reg_index);

```
/*
Turn Off ('0') values of a specifc register:
  - reg_index: specify a register index.

ex: myShiftRegister.OffRegister(1);
*/
```

boolean OffAllRegisters();

```
/*
Turn Off ('0') values of all registers.

ex: myShiftRegister.OffAllRegisters();
*/
```

boolean OnRegister(int reg_index);

```
/*
turn On ('1') values of a specifc register:
  - reg_index: specify a register index.

ex: myShiftRegister.OnRegister(1);
*/
```

boolean OnAllRegisters();

```
/*
Turn On ('1') values of all registers.

ex: myShiftRegister.OnAllRegisters();
*/
```

boolean OnSingleRegister(int position, int reg_index);

```
/*
Turn On ('1') value at a specific position of a specific register:

  - position:  position of the value to turn on.
                position must be beetween 0 and 7.
  - reg_index: specify a register index.

ex: myShiftRegister.OnSingleRegister(3, 1);
*/
```

boolean OnSingleAllRegisters(int position);

```
/*
Turn On ('1') value at a specific position among all registers:

    - position:    position of the value to turn on.
                   position must be beetween 0 and 8 x number_of_registers-1.

ex: myShiftRegister.OnSingleAllRegisters(12);
*/
```

boolean OnSingleEachRegisters(int position);

```
/*
Turn On ('1') value at a specific position for each register:

    - position:    position of the value to turn on.
                   position must be beetween 0 and 7.

ex: myShiftRegister.OnSingleEachRegisters(4);
*/
```

boolean OffSingleRegister(int position, int reg_index);

```
/*
Turn Off ('0') value at a specific position of a specific register:

    - position:    position of the value to turn off.
                   position must be beetween 0 and 7.
    - reg_index:   specify a register index.

ex: myShiftRegister.OffSingleRegister(3, 1);
*/
```

boolean OffSingleAllRegisters(int position);

```
/*
Turn Off ('0') value at a specific position among all registers:

    - position:    position of the value to turn off
                   position must be beetween 0 and 8 x number_of_registers-1.

ex: myShiftRegister.OffSingleAllRegisters(12);
*/
```

boolean OffSingleEachRegisters(int position);

```
/*
Turn Off ('0') value at a specific position for each register:

    - position:    position of the value to turn off.
                   position must be beetween 0 and 7.

ex: myShiftRegister.OffSingleEachRegisters(4);
*/
```

boolean OnSingleOnlyRegister(int position, int reg_index);

```
/*
Turn On ('1') value at a specific position of a specific register,
and Turn Off all other values for this register:
```

- position: position of the value to turn on.
position must be between 0 and 7.
- reg_index: specify a register index.

```
ex: myShiftRegister.OnSingleOnlyRegister(3, 1);  
*/
```

boolean OnSingleOnlyAllRegisters(int position);

```
/*  
Turn On ('1') value at a specific position among all registers,  
and Turn Off all other values:  
  
- position: position of the value to turn on.  
position must be between 0 and 8 x number_of_registers-1.  
  
ex: myShiftRegister.OnSingleOnlyAllRegisters(12);  
*/
```

boolean OnSingleOnlyEachRegisters(int position);

```
/*  
Turn On ('1') value at a specific position for each register,  
and Turn Off all other values:  
  
- position: position of the value to turn on.  
position must be between 0 and 7.  
  
ex: myShiftRegister.OnSingleOnlyEachRegisters(4);  
*/
```

boolean OffSingleOnlyRegister(int position, int reg_index);

```
/*  
Turn Off ('0') value at a specific position of a specific register,  
and Turn On all other values for this register:  
  
- position: position of the value to turn off.  
position must be between 0 and 7.  
- reg_index: specify a register index.  
  
ex: myShiftRegister.OffSingleOnlyRegister(3, 1);  
*/
```

boolean OffSingleOnlyAllRegisters(int position);

```
/*  
Turn Off ('0') value at a specific position among all registers,  
and Turn On all other values:  
  
- position: position of the value to turn off  
position must be between 0 and 8 x number_of_registers-1.  
  
ex: myShiftRegister.OffSingleOnlyAllRegisters(12);  
*/
```

boolean OffSingleEachRegisters(int position);

```
/*  
Turn Off ('0') value at a specific position for each register,  
and Turn On all other values:
```

- position: position of the value to turn off.
position must be between 0 and 7.

```
ex: myShiftRegister.OffSingleOnlyEachRegisters(4);  
*/
```

boolean ShiftLeftRegister(boolean circular, int reg_index);

```
/*  
Shift to the left values of a specific register:  
  
- circular: set whether the shift is circular.  
- reg_index: specify a register index.  
  
ex: // before: [1,0,0,1,1,0,0,0]  
    myShiftRegister.ShiftLeftRegister(true, 1);  
    // after:  [0,0,1,1,0,0,0,1]  
  
ex: // before: [1,0,0,1,1,0,0,0]  
    myShiftRegister.ShiftLeftRegister(true, 1);  
    // after:  [0,0,1,1,0,0,0,0]  
*/
```

boolean ShiftLeftAllRegisters(boolean circular);

```
/*  
Shift to the left values for all registers:  
  
- circular: set whether the shift is circular.  
  
ex: // before: [1,0,0,1,1,0,0,0][1,1,0,0,0,0,0,0]  
    myShiftRegister.ShiftLeftRegister(true);  
    // after:  [0,0,1,1,0,0,0,1][1,0,0,0,0,0,0,1]  
*/
```

boolean ShiftLeftEachRegisters(boolean circular);

```
/*  
Shift to the left values for each registers:  
  
- circular: set whether the shift is circular.  
  
ex: // before: [1,0,0,1,1,0,0,0][0,1,1,1,0,0,0,0]  
    myShiftRegister.ShiftLeftRegister(true);  
    // after:  [0,0,1,1,0,0,0,1][1,1,1,0,0,0,0,0]  
*/
```

boolean ShiftRightRegister(boolean circular, int reg_index);

```
/*  
Shift to the right values of a specific register:  
  
- circular: set whether the shift is circular.  
- reg_index: specify a register index.  
  
ex: // before: [1,0,0,1,1,0,0,1]  
    myShiftRegister.ShiftRightRegister(true, 1);  
    // after:  [1,1,0,0,1,1,0,0]  
  
ex: // before: [1,0,0,1,1,0,0,1]  
    myShiftRegister.ShiftLeftRegister(true, 1);  
    // after:  [0,1,0,0,1,1,0,0]  
*/
```

boolean ShiftRightAllRegisters(boolean circular);

```
/*
Shift to the right values for all registers:

    - circular:  set whether the shift is circular.

ex: // before: [1,0,0,1,1,0,0,0][1,1,0,0,0,0,0,1]
    myShiftRegister.ShiftLeftRegister(true);
    // after:  [1,1,0,0,1,1,0,0][0,1,1,0,0,0,0,0]
*/
```

boolean ShiftRightEachRegisters(boolean circular);

```
/*
Shift to the right values for each registers:

    - circular:  set whether the shift is circular.

ex: // before: [1,0,0,1,1,0,0,0][0,1,1,1,0,0,0,1]
    myShiftRegister.ShiftLeftRegister(true);
    // after:  [0,1,0,0,1,1,0,0][1,0,1,1,1,0,0,0]
*/
```

boolean NegateRegister(int reg_index);

```
/*
Negate values of a specific register. On values are turned Off,
Off values are turned On:

    - reg_index: specify a register index.

ex: myShiftRegister.NegateRegister(1);
*/
```

boolean NegateAllRegisters();

```
/*
Negate values of all registers.

ex: myShiftRegister.NegateAllRegisters();
*/
```

boolean Update();

```
/*
Update register state. Register output pins are updated
to their current values.

ex: myShiftRegister.Update();
*/
```

boolean Blink(int iter, int delay_val);

```
/*
Update Off then update to their current values the
output pins of the registers.

    - iter:  number of blinks.
    - delay_val: blink delay in milliseconds.
```

```
ex: myShiftRegister.Blink(5, 100);  
*/
```

boolean TestSequence();

```
/*  
Run a test sequence for N cascaded registers with LEDs.  
  
- Turn On all LEDs  
- Blink LEDs 5 times @ 100ms  
- Turn Off all LEDs  
- Turn On 3 LEDs  
- Shift the 3 LEDs positions to the left x times  
- Turn On / Off groups of 8 LEDs in alternance  
- Switch On / Off LED states in alternance  
  
ex: myShiftRegister.TestSequence();  
*/
```

Shift register parameters::

boolean verbose;

```
/*  
Enable/Disable verbose output.  
When enabled, each operation output a string with its name.  
Default value is false.  
  
ex: myShiftRegister.verbose = true  
*/
```

boolean auto_update;

```
/*  
Enable/Disable register state auto update.  
When enabled, each action updates values of the registers' output pins,  
then wait some amount of time defined by the update_delay value.  
When disabled, registers' output pins remain unchanged until  
an explicit call to Update().  
Default value is true.  
  
ex: myShiftRegister.auto_update = false;  
    myShiftRegister.OffAllRegisters();  
    myShiftRegister.OnSingleAllRegisters(7);  
    myShiftRegister.OnSingleAllRegisters(12);  
    myShiftRegister.Update();  
    delay(1000);  
*/
```

int update_delay;

```
/*  
Set the amount of time, in milliseconds, an action auto update  
is maintained before returning.  
Default value is 0.  
  
ex: myShiftRegister.update_delay = 250;  
*/
```

