Interface Shift Register with Atmega

74HC595 (8Bits)

Preface

- When there more devices to drive than available MCU pins, this shift register comes handy
- Essentially a serial to parallel converter
- Typical applications include driving 7 Segment display, Bargraph display etc.
- One or more shift registers can be cascaded with the existing one (daisy chaining) to get 16 or 32 or more separate output lines without having to increase MCU pins
- These work as output devices. There are devices available for input as well

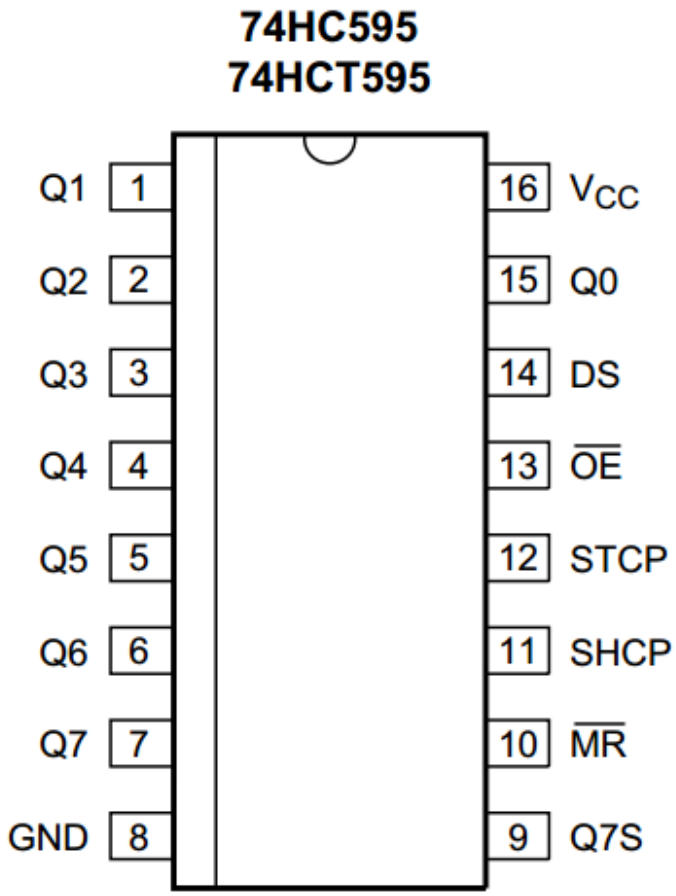
Refference

- Datasheet: E:\educational\datasheets\74HC_HCT595.pdf

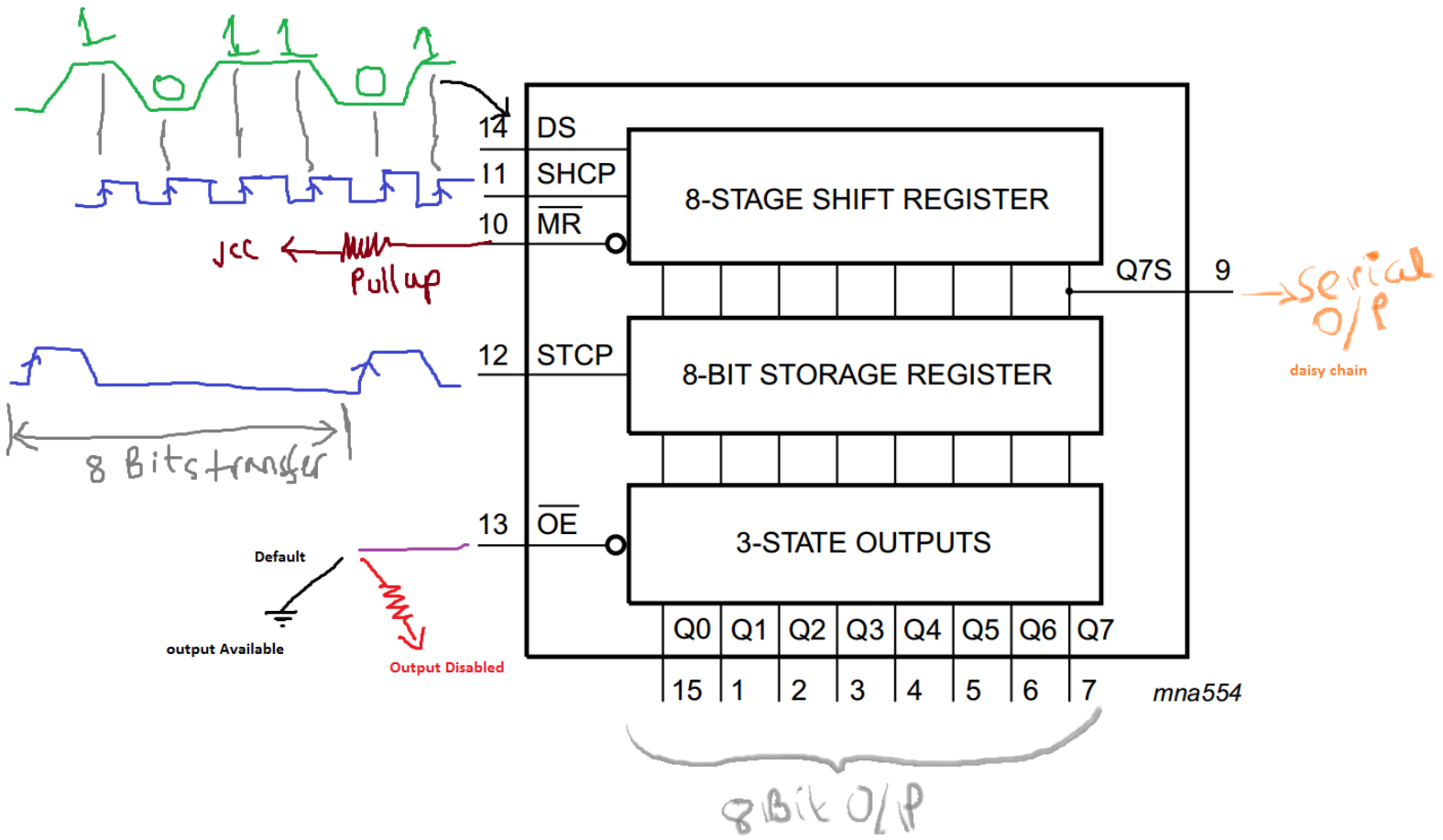
- Extreme-Electronics: http://extremeelectronics.co.in/avr-tutorials/using-shift-registers-with-avr-micro-avr-tutorial/

- Youtube : https://www.youtube.com/watch?v=d7Au3I4ZdZc

Pin out


74HC595
74HCT595
Q1 1
Q2 2
Q3 3
Q4 4
Q5 5
Q6 6
Q7 7
GND 8
16 VCC
15 Q0
14 DS
13 OE
12 STCP
11 SHCP
10 MR
9 Q7S

Working mechanism



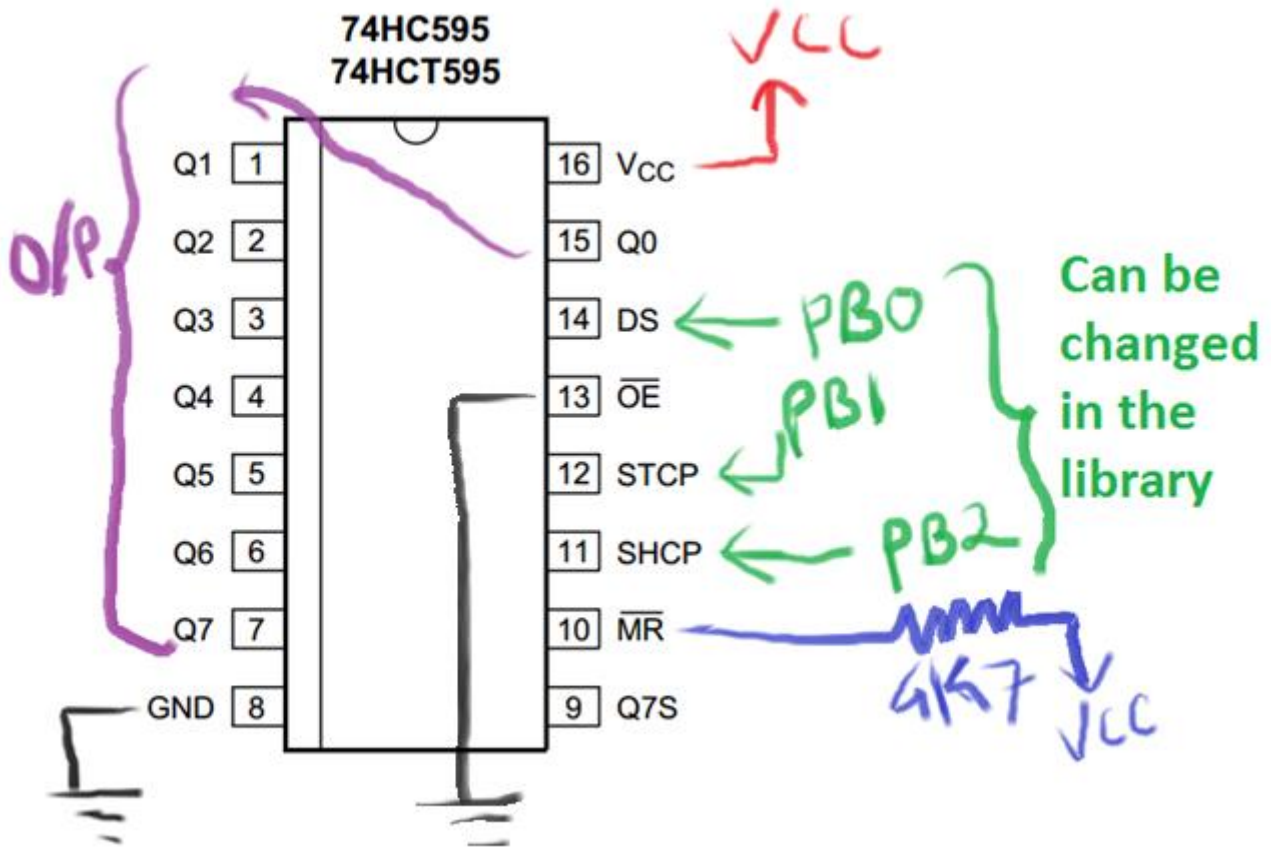
Working Principle DS and SHCP

- DS (serial data pin-14) takes in serial data.
- At every rising edge of shift clk on pin 11, the register left-shifts its previous byte present in 8 STAGE SHIFT REGISTER.
- samples and captures DS (high or low) and places it on newly vacant LSB position
- This way 8 rising edge on SHCP will load a new byte in 8 STAGE SHIFT REGISTER.

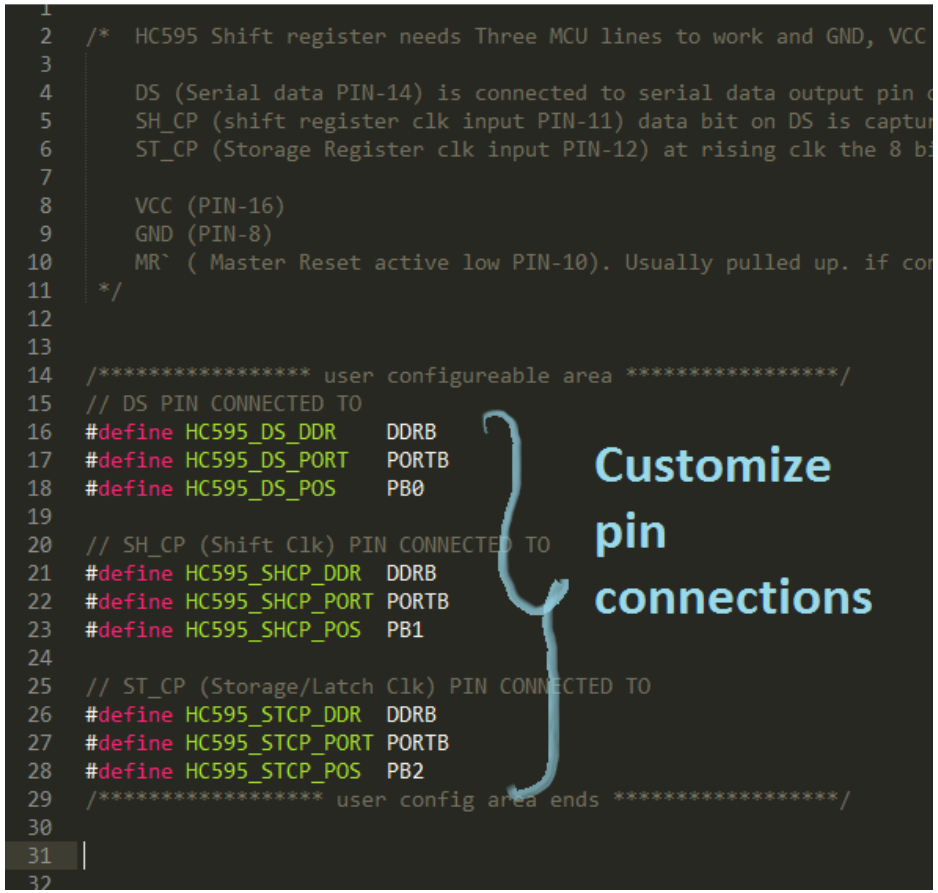
Working Principle STCP

- Unless STCP receive a rising edge the content of 8 STAGE SHIFT REGISTER will not be available at 8 BIT STORAGE REGISTER
- Once STCP receives a rising edge the content of 8 STAGE SHIFT REGISTER gets copied into 8 BIT STORAGE REGISTER
- So its usually done once after 8 new bits have shifted in 8 STAGE SHIFT REGISTER
- When OE is connected to GND (default) the 8 bits in 8 BIT STORAGE REGISTER is available in 3 STATE OUTPUTS for external use

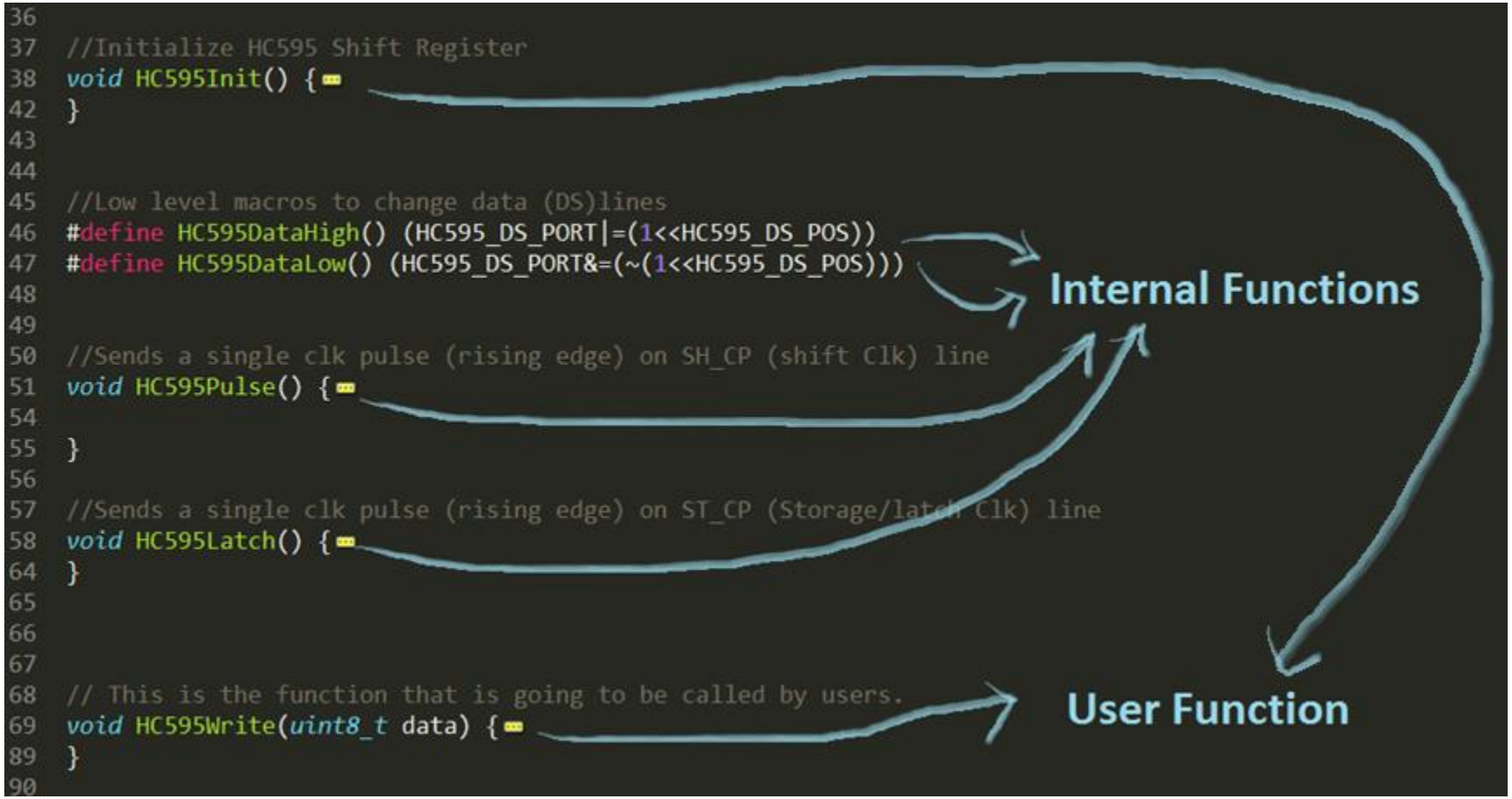
MCU connection (Basic mode) just 5 pins (3MCU pins)



Library (ShiftRegHC595.h)



Library (ShiftRegHC595.h)



Library (ShiftRegHC595.h)

```
100  #include <avr/io.h>
101  #include <util/delay.h>
102
103  #include "ShiftRegHC595.h"
104
105
106
107  void main(){
108
109      //Initialize HC595 system
110      HC595Init();
111      while(1) {
112          HC595Write(0b00110101); //Write the data to HC595
113          _delay_ms(500);                 //Wait
114          HC595Write(0b11001010);
115          _delay_ms(500);
116      }
117  }
118
119
120
```

Fully running code Example

Extended & Advanced Features

- Sometimes one shift register is not enough.
- So there are four different ways to connect multiple shift registers with MCU
  1. Daisy Chain
  2. Common clocks(SHCP & STCP common and different DS lines)
  3. Common latch (STCP common and different SHCP and DS lines)
  4. Totally separate modules

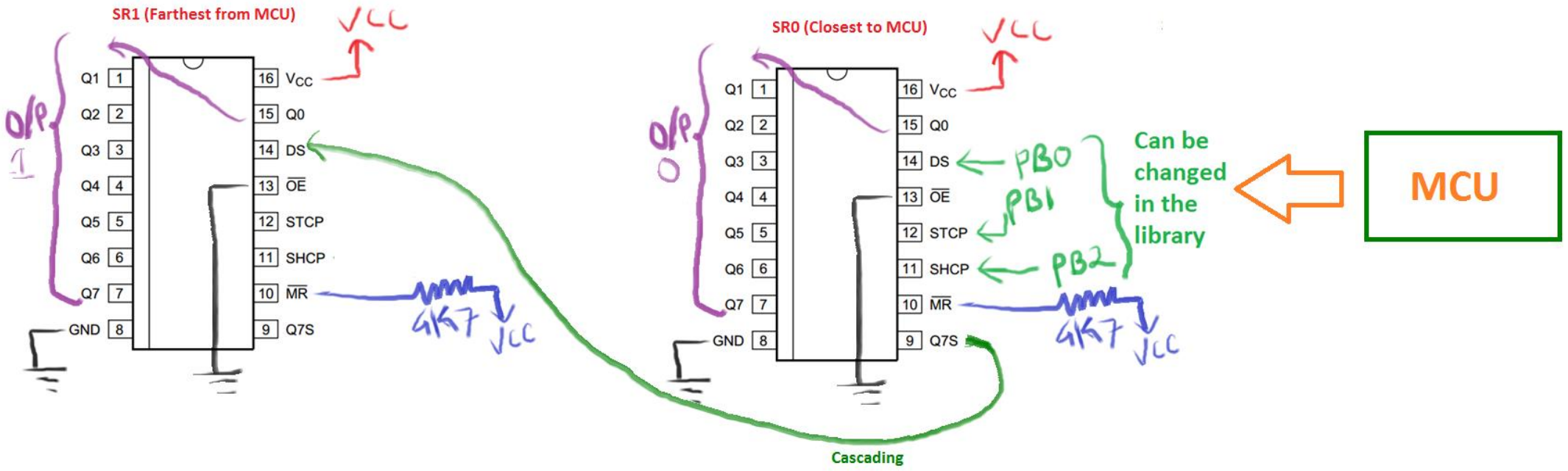
1. Daisy Chain

Cascade

1. Daisy Chain

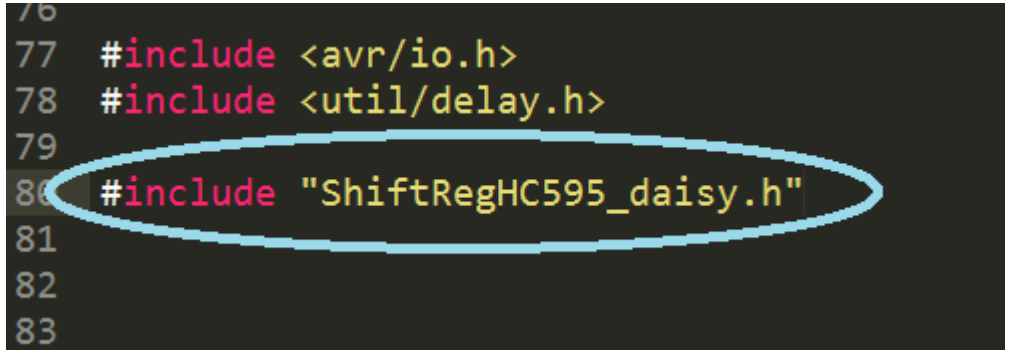
- Shift registers are cascaded.
- Number of MCU lines doesn't increase(stays 3).
- 2 SR gives 16 o/p lines, 3 gives 24 etc.
- But total number of bits avail able have to be updated together
- Best suited for display devices (7 segments, bargraphs etc.)

1.Daisy chain Connections



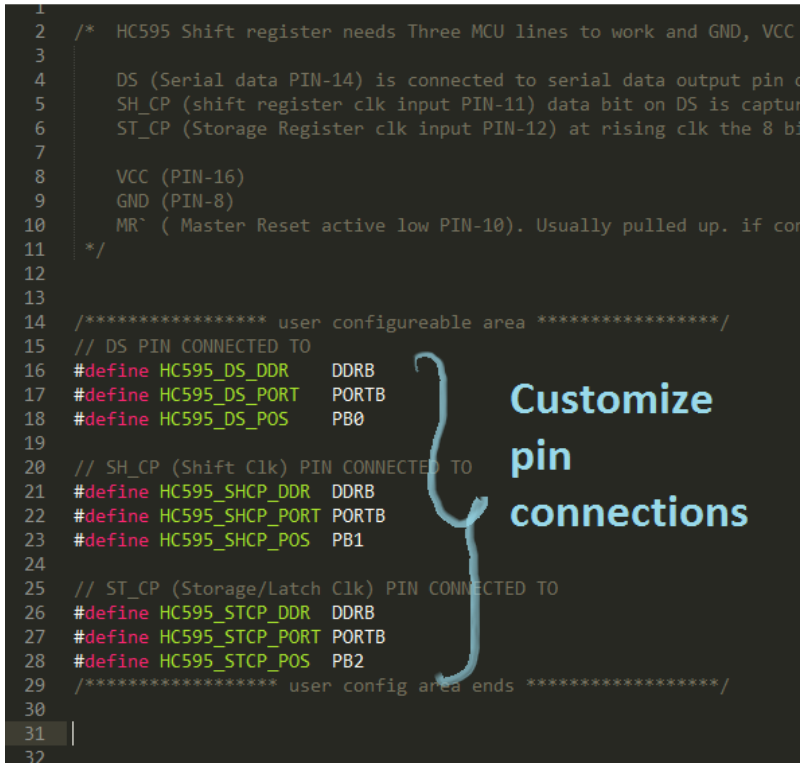
1.Daisy chain Library (shiftreghc595_daisy.h)

- Include “ShiftRegHC595_daisy.h” and this will automatically include “ShiftRegHC595.h”



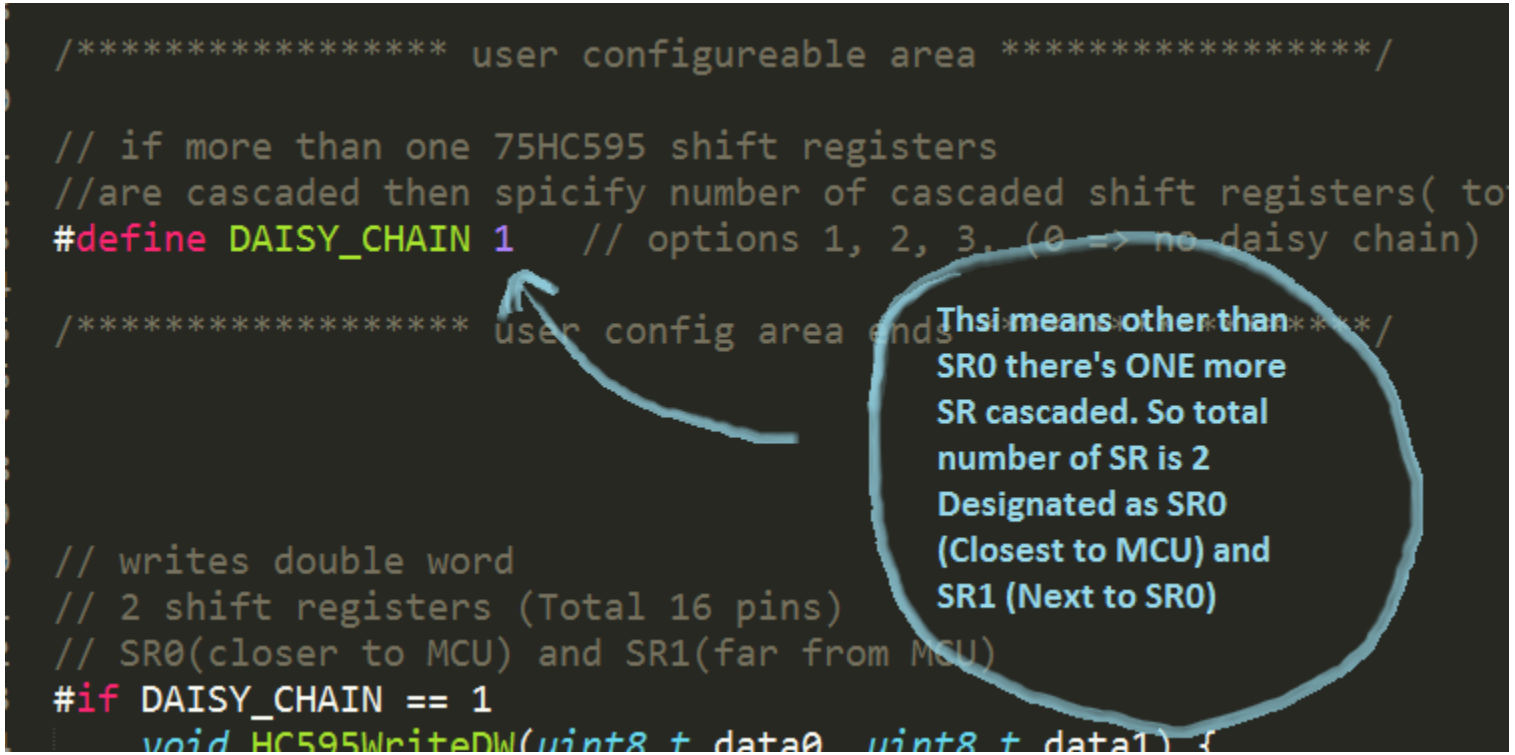
1.Daisy chain Library (shiftreghc595_daisy.h)

- Then configure the connection for SR0 (Closest to MCU) in ShiftRegHC595.h



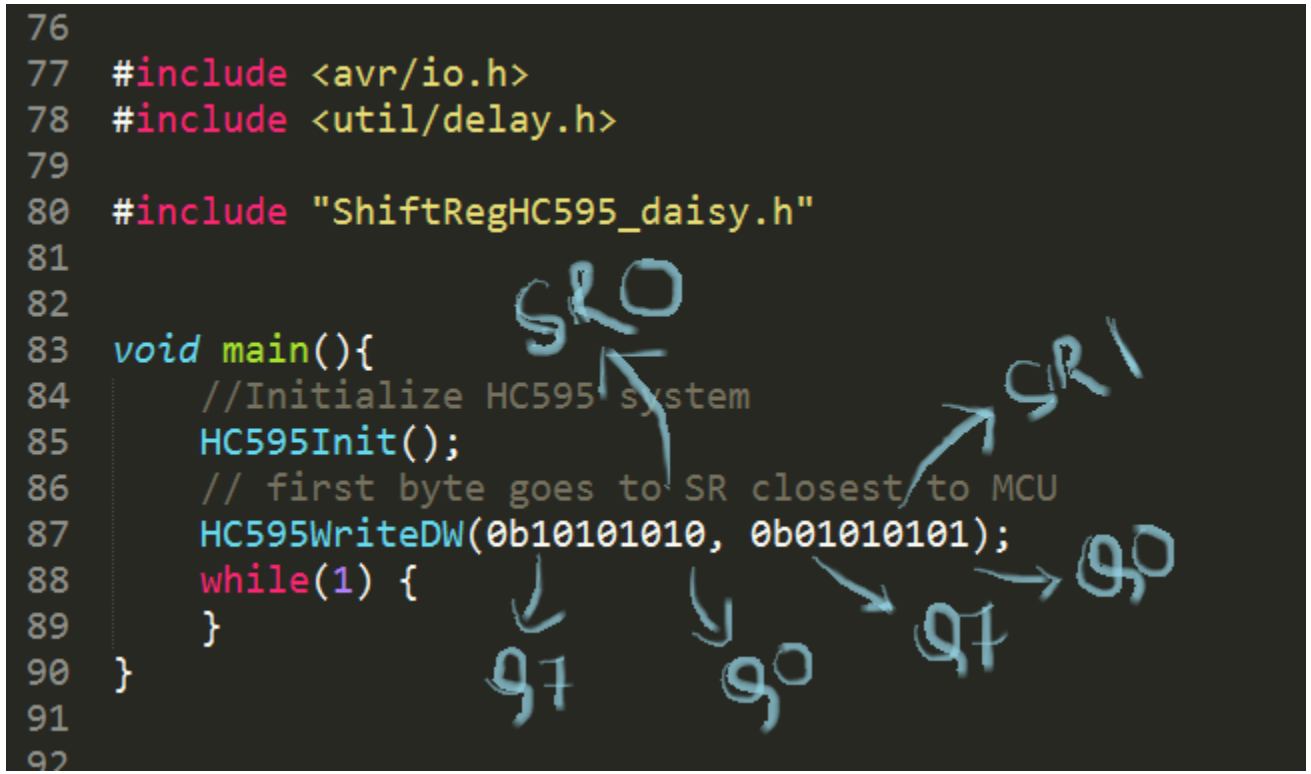
1.Daisy chain Library (shiftreghc595_daisy.h)

- Then specify number of SR(s) cascaded, i.e. total number of SR(s) excluding SR0, in ShiftRegHC595_daisy.h



1.Daisy chain Library (shiftreghc595_daisy.h)

- That’s it, Now just call the functions. Here’s a sample program.



```
76
77  #include <avr/io.h>
78  #include <util/delay.h>
79
80  #include "ShiftRegHC595_daisy.h"
81
82
83  void main(){
84      //Initialize HC595 system
85      HC595Init();
86      // first byte goes to SR closest to MCU
87      HC595WriteDW(0b10101010, 0b01010101);
88      while(1) {
89      }
90  }
91
92
```

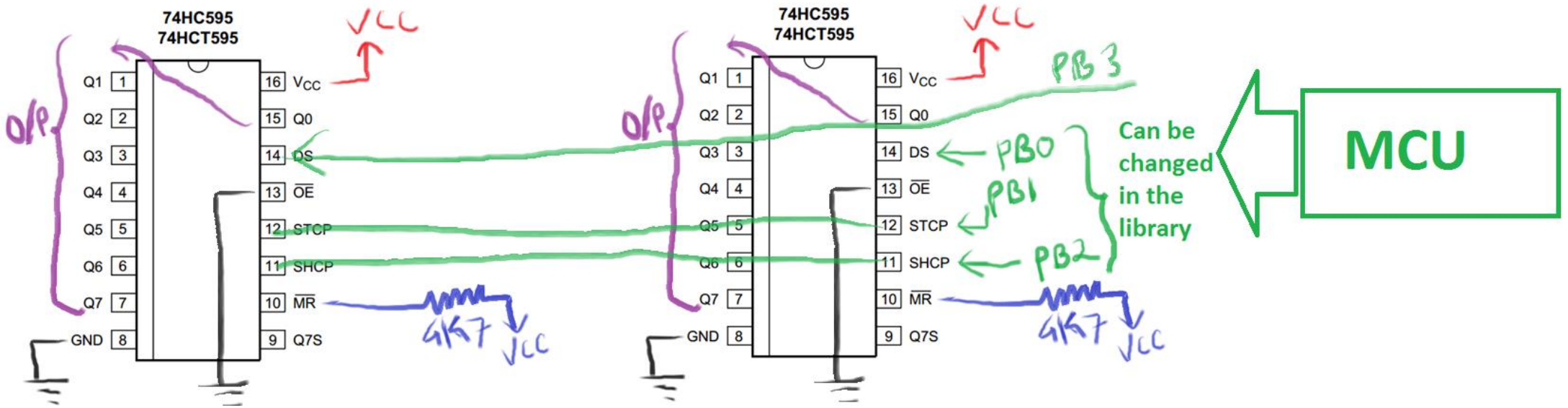
2. Common Clocks

SHCP(shift Clk) & STCP(latch Clk)

2. Common Clocks (SHCP & STCP)

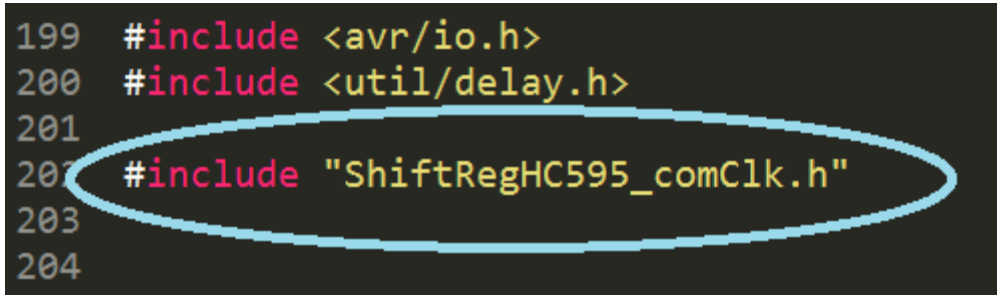
- Shift registers are not cascaded.
- All the SR(s) share the same clock line (SHCP & STCP)
- But each SR have their own an different DS lines
- Since they share clocks. If one SR is updated then this will cause shifts in all the SR(s). To overcome this all shift registers will have to be updated all at once.
- First SR (SR0) requires 3 MCU lines and then after one more MCU lines (DS) each for each SR(s) added.

2. Common Clocks Connections



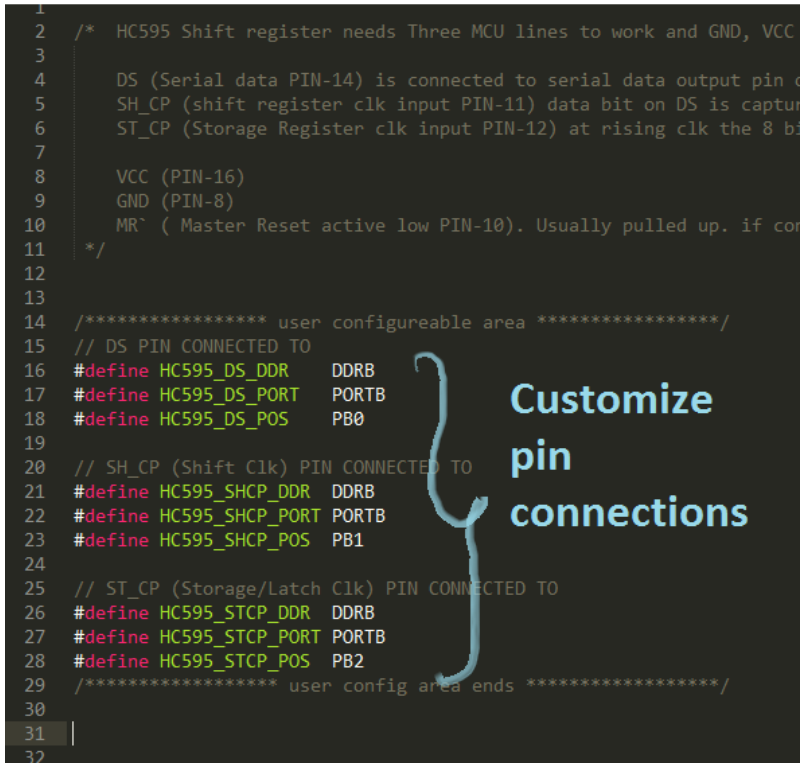
2. Common Clocks Library (shiftreghc595_comClk.h)

- Include “ShiftRegHC595_comClk.h” and this will automatically include “ShiftRegHC595.h”



2. Common Clocks Library (shiftreghc595_comClk.h)

- Then configure the connection for SR0 (Closest to MCU) in ShiftRegHC595.h



2. Common Clocks Library (shiftreghc595_comClk.h)

- Then configure DSx line connections for SR1 and SR2 in ShiftRegHC595_comClk.h

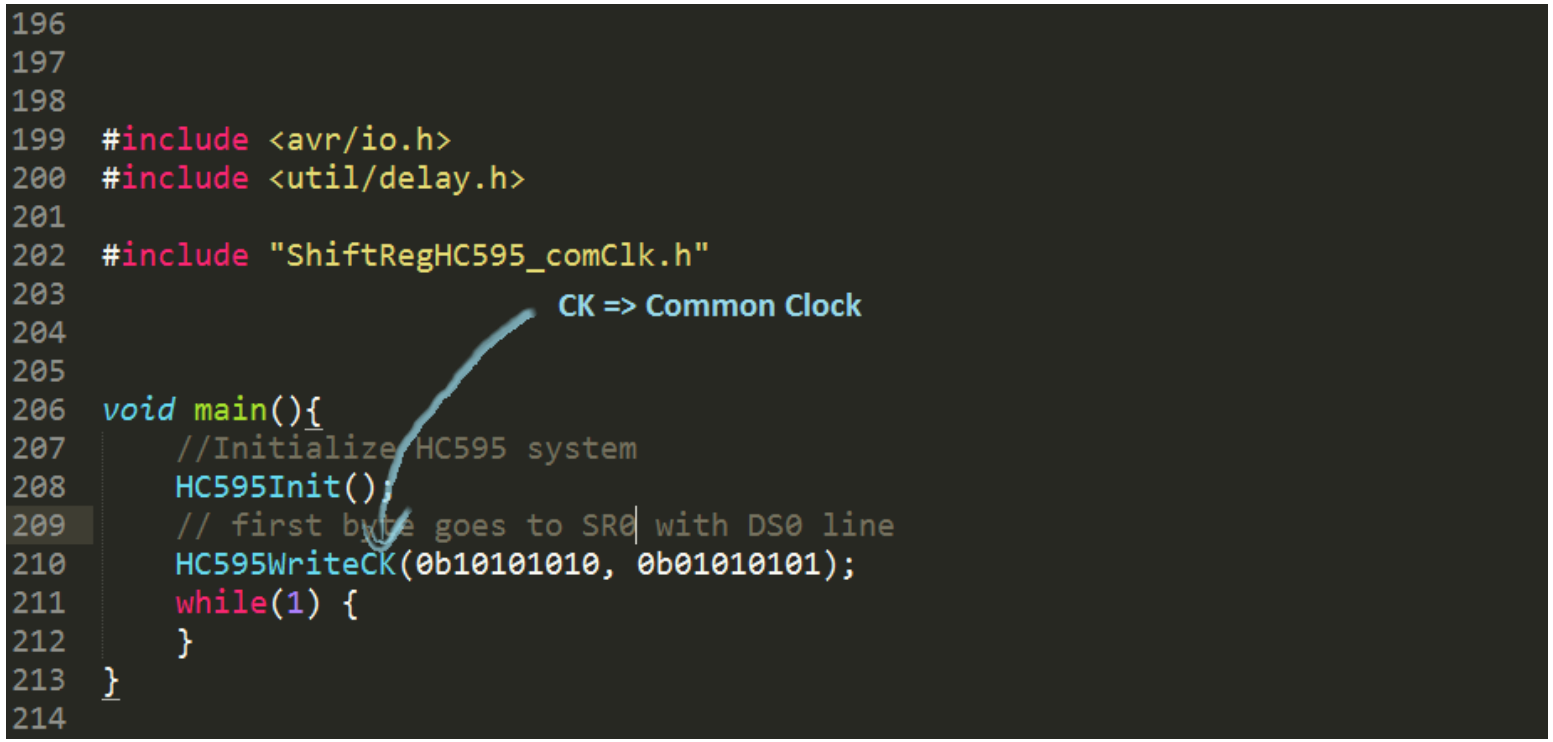
```
// If more than one shift registers are connected (NOT CASCADED)
// They can and will share same "shift clk" and "latch clock" lines
// But serial data lines will differ. i.e. DS1, DS2, DS3 etc.
// In such case specify the aditional data lines here.


// DS1 PIN CONNECTED TO
#define HC595_DS1_DDR    DDRB
#define HC595_DS1_PORT   PORTB
#define HC595_DS1_POS    PB3

// // DS2 PIN CONNECTED TO
// #define HC595_DS2_DDR    DDRB
// #define HC595_DS2_PORT   PORTB
// #define HC595_DS2_POS    PB4


// ... you can go on and continue to add more shift registers if needed

/****************** user config area ends ******************/
```

DS1 connection for SR1

If another SR's needed then uncomment and configure

2. Common Clocks Library (shiftreghc595_comClk.h)

- That’s it, Now just call the functions. Here’s a sample program.




```
196
197
198
199  #include <avr/io.h>
200  #include <util/delay.h>
201
202  #include "ShiftRegHC595_comClk.h"
203
204
205
206  void main(){
207      //Initialize HC595 system
208      HC595Init();
209      // first byte goes to SR0 with DS0 line
210      HC595WriteCK(0b10101010, 0b01010101);
211      while(1) {
212      }
213  }
214
```

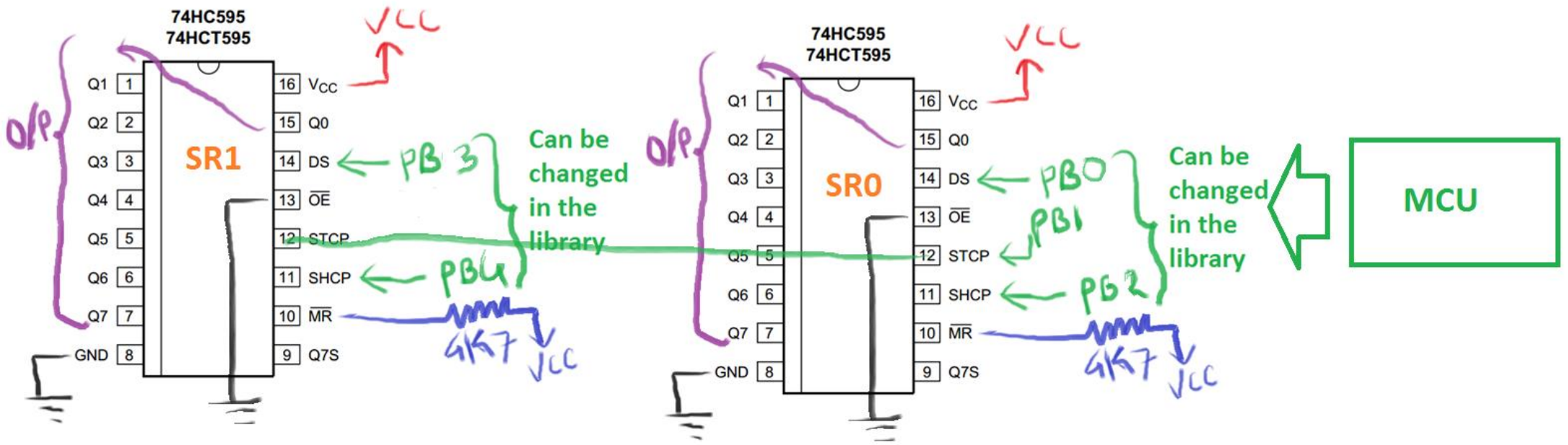
3. Common latch

STCP(latch Clk is common)

3. Common Latch

- The latch Clk(STCP) is shared between all the SR(s)
- Each SR has its own Shift Clk(SHCP) and Serial data(DS)
- This mode can be achieved by Configuring the same PORT, DDR, POS for STCP pic in Totally separate mode

3. Common Latch Connections



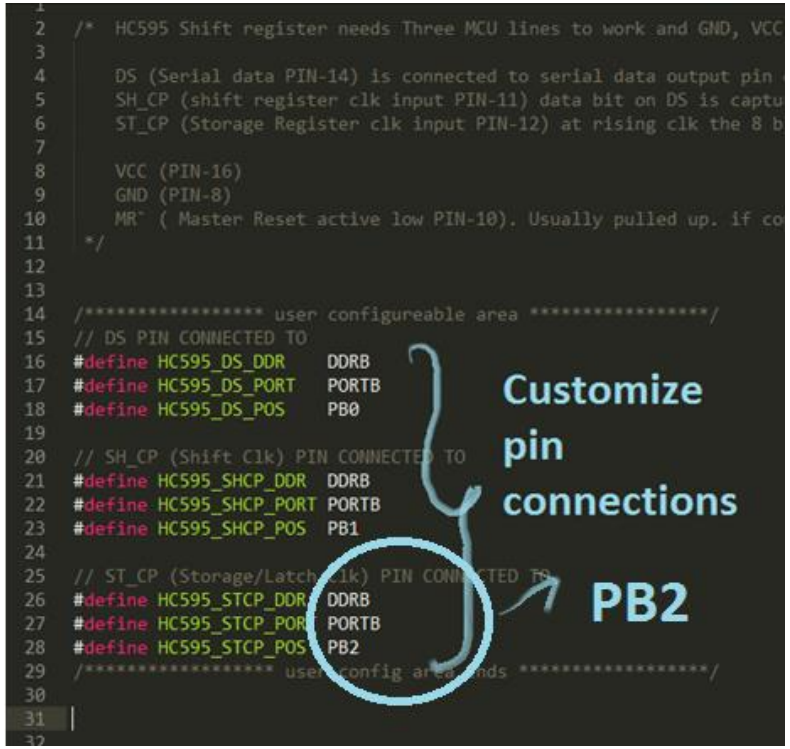
3. Common Latch Library (ShiftRegHC595_X.h)

- Include “ShiftRegHC595_X.h” and this will automatically include “ShiftRegHC595.h”

```
180
181  #include <avr/io.h>
182  #include <util/delay.h>
183
184  #include "ShiftRegHC595_X.h"
185
186
187
```

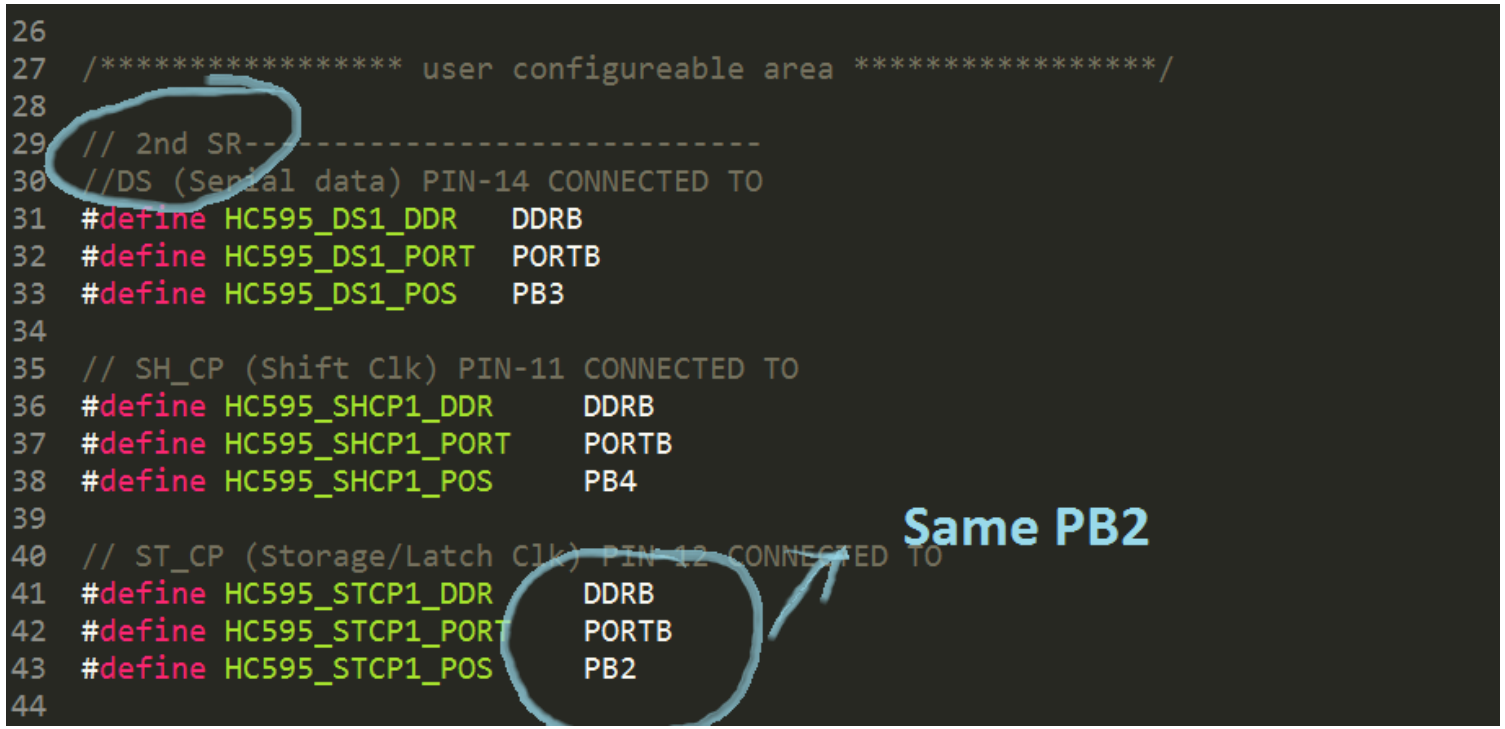
3. Common Latch Library (ShiftRegHC595_X.h)

- Then configure the connection for SR0 (Closest to MCU) in ShiftRegHC595.h



3. Common Latch Library (ShiftRegHC595_X.h)

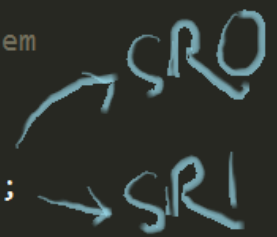
- Then configure the connection for SR1 in ShiftRegHC595_X.h



3. Common Latch Library (ShiftRegHC595_X.h)

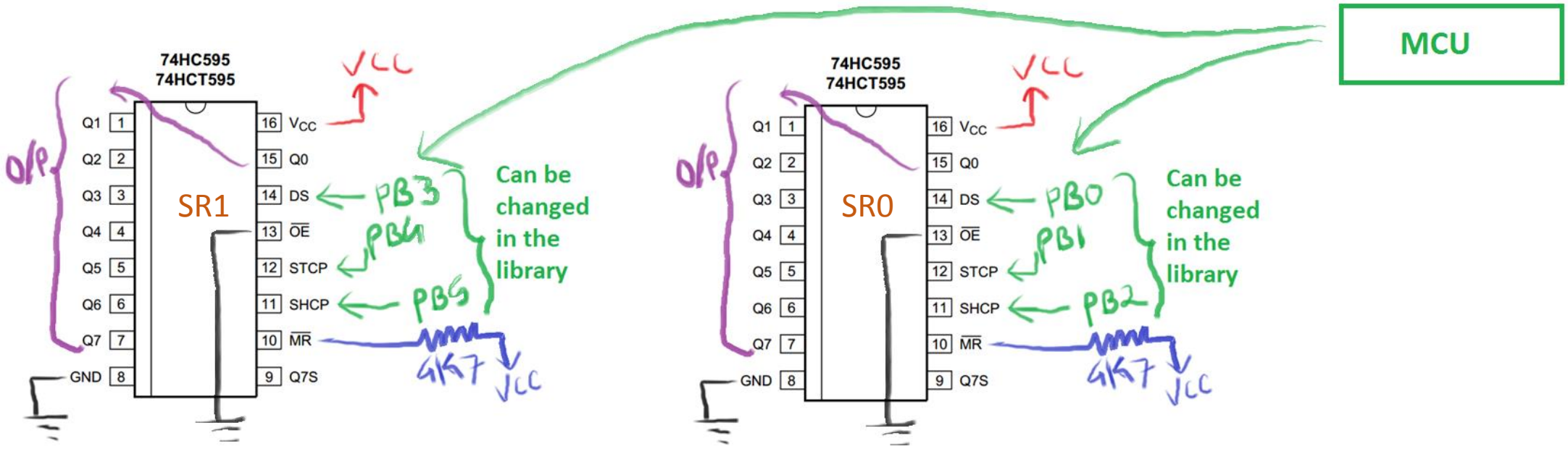
- Sample Program

```
179
180
181  #include <avr/io.h>
182  #include <util/delay.h>
183
184  #include "ShiftRegHC595_X.h"
185
186
187
188  void main(){
189      //Initialize HC595 system
190      HC595Init();
191      HC595Init1();
192      HC595Write(0b10101010);
193      HC595Write1(0b11001100);
194      while(1) {
195      }
196  }
197
198
199
200
```



4. Totally Separate

4. Totally separate Connections



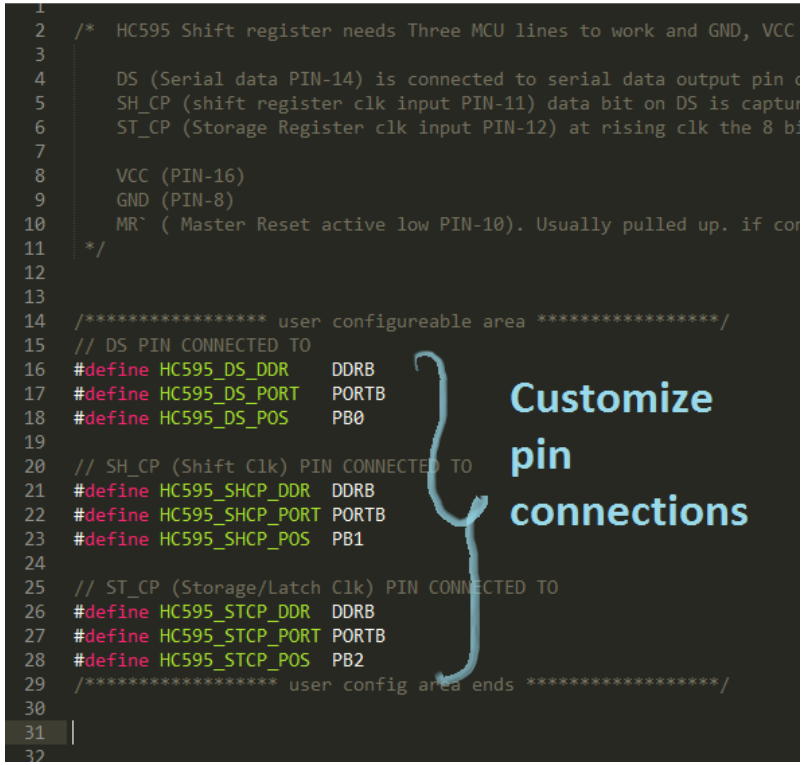
4. Totally separate Connections

- Include “ShiftRegHC595_X.h” and this will automatically include “ShiftRegHC595.h”

```
180
181  #include <avr/io.h>
182  #include <util/delay.h>
183
184  #include "ShiftRegHC595_X.h"
185
186
187
```

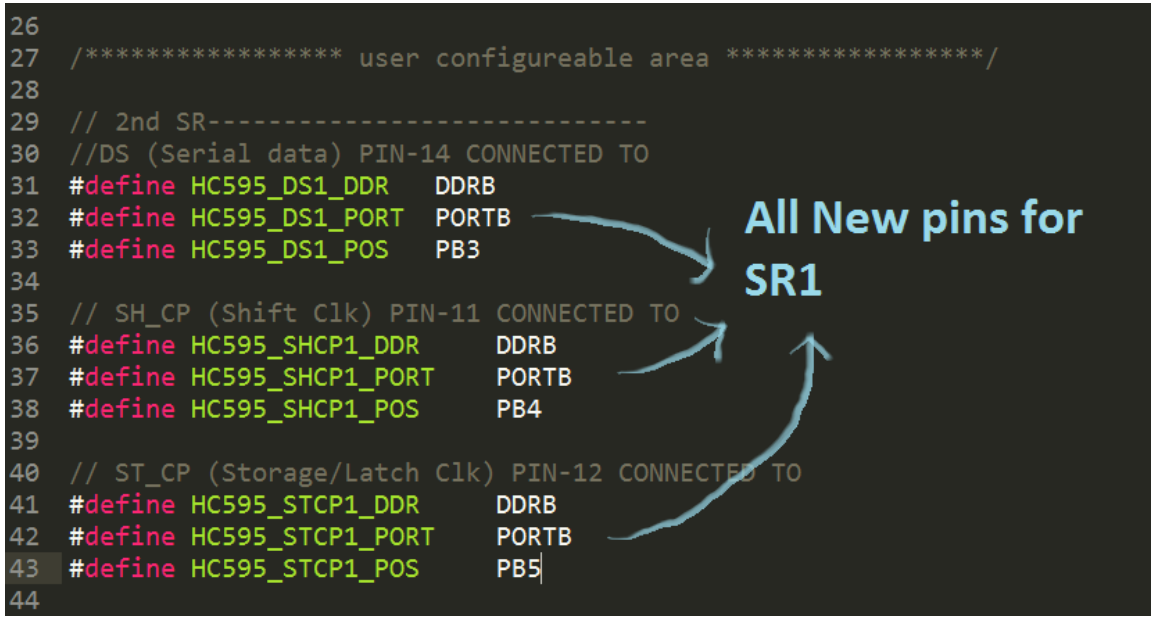
4. Totally separate Connections

- Then configure the connection for SR0 (Closest to MCU) in ShiftRegHC595.h



4. Totally separate Connections

- Then configure the connection for SR1 in ShiftRegHC595_X.h



4. Totally separate Connections

- Sample Program

```
179
180
181  #include <avr/io.h>
182  #include <util/delay.h>
183
184  #include "ShiftRegHC595_X.h"
185
186
187
188  void main(){
189      //Initialize HC595 system
190      HC595Init();
191      HC595Init1();
192      HC595Write(0b10101010);
193      HC595Write1(0b11001100);
194      while(1) {
195      }
196  }
197
198
199
200
```