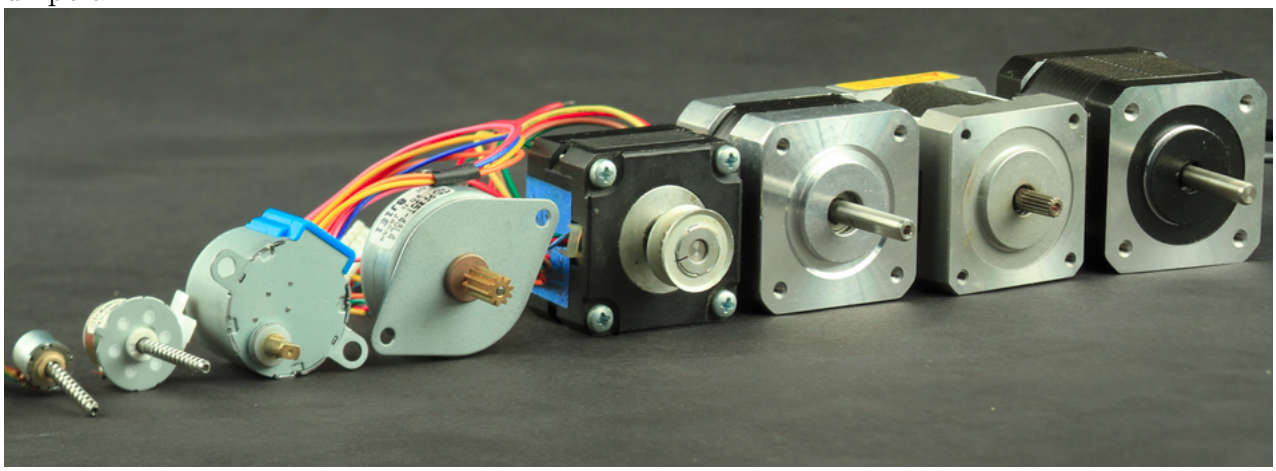
Stepper Motor

Yinchheanyun

2 November 2023

1. **What is Stepper Motor?**
   Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time.
   What are stepper good for?
   A Stepper motor has maximum torque at low speeds, so they are a good choice for applications requiring low speed with high precision.

2. **Types of Steppers**
   These are: Permanent Magnet or Hybrid steppers, either 2-phase bipolar, or 4-phase unipolar.



- Motor Size
  - Stepper motors come in sizes ranging from smaller than a peanut to big NEMA 57 monsters.
  - Most motors have torque ratings. This is what you need to look at to decide if the motor has the strength to do what you want.
  - NEMA 17 is a common size used in 3D printers and smaller CNC mills. Smaller motors find applications in many robotic and animatronic applications.
  - The NEMA numbers define standard faceplate dimensions for mounting the motor.
- Step Count
  - The number of steps per revolution ranges from 4 to 400.
  - Commonly available step counts are 24, 48 and 200.
  - Resolution is often expressed as degrees per step. A 1.8° motor is the same as a 200 step/revolution motor.

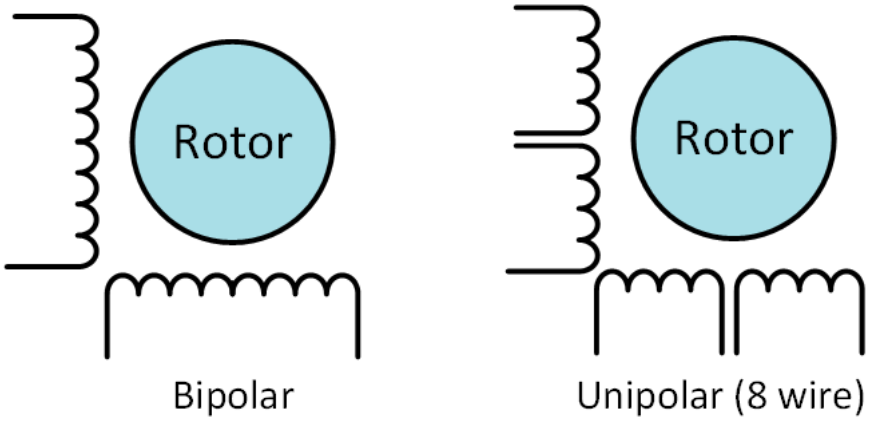
  - High step count motors top-out at lower RPMs than similar size.
  - The higher step-rates needed to turn these motors results in lower torque than a similar size low-step-count motor at similar speeds.

- Gearing
  - Another way to achieve high positioning resolution is with gearing. A 32:1 gear-train applied to the output of an 8-steps/revolution motor will result in a 256 step motor.
  - A gear train will also increase the torque of the motor. Some tiny geared steppers are capable of impressive torque. But the tradeoff of course is speed. Geared stepper motors are generally limited to low RPM applications.
  - Backlash is another issue with geared motors. When the motor reverses direction, it needs to take up any slack there may be in the gear-train. This can affect positioning accuracy.

- Shaft Style
  Motors are available with a number of shaft styles:
  - Round or "D" Shaft: These are available in a variety of standard diameters and there are many pulleys, gears and shaft couplers designed to fit. "D" shafts have one flattened side to help prevent slippage. These are desirable when high torques are involved.
  - Geared shaft: Some shafts have gear teeth milled right into them. These are typically designed to mate with modular gear trains.
  - Lead-Screw Shaft: Motors with lead-screw shafts are used to build linear actuators. Miniature versions of these can be found as head positioners in many disk drives.

- Wiring
  These are Permanent Magnet or Hybrid steppers wired as 2-phase bipolar, or 4-phase unipolar.

- Coils and Phases
  - A stepper motor may have any number of coils.
  - These are connected in groups called "phases".

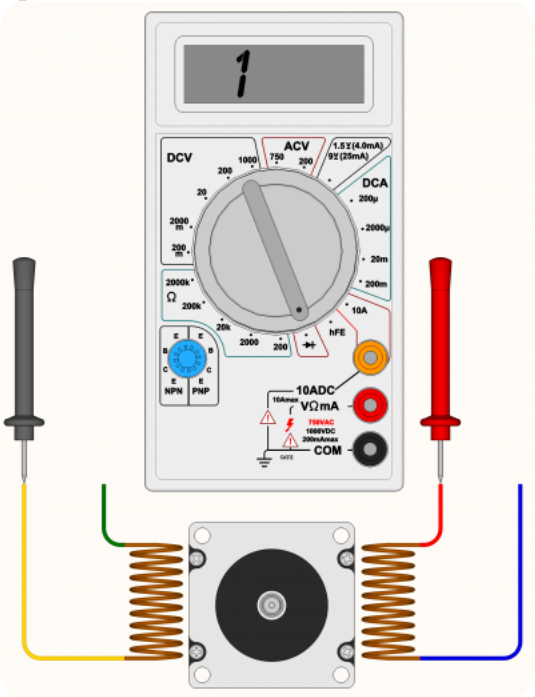


- Unipolar vs. Bipolar
  - Unipolar drivers, always energize the phases in the same way. One lead, the "common" lead, will always be negative. The other lead will always be positive. Unipolar drivers can be implemented with simple transistor circuitry. The disadvantage is that there is less available torque because only half of the coils can be energized at a time.

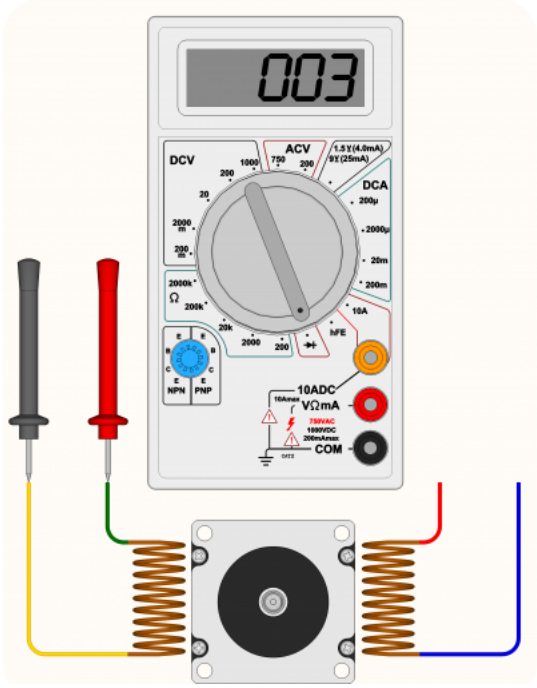
– Bipolar drivers use H-bridge circuitry to actually reverse the current flow through the phases. By energizing the phases with alternating the polarity, all the coils can be put to work turning the motor.

3. **How to identify the motor coil wiring pairs**

- Set the Multimeter to the diode test position in
- The Multimeter will not make any sound, and the display will still be "1" so the coil not pair.



- when The Multimeter may sound a soft squeak, and a different number will appear on the display instead of the digit one.



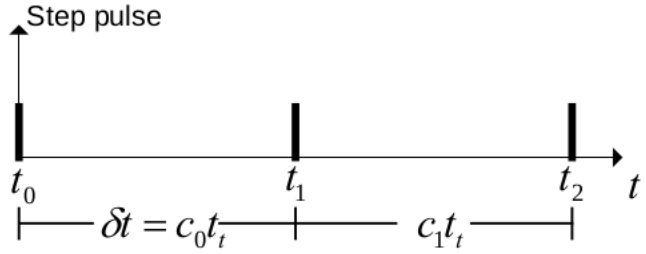
Separate them from the others and connect to the connectors of the stepper driver A+/A- and B+/B-.

4. **How to Control Stepper_Motor**

4.1 **Fundamental stepper motor equations**

To rotate the stepper motor at a constant speed, pulses must be generated at a steady rate, shown in Figure:

Figure 2-4. Stepper motor pulses



- A counter generates these pulses, running at the frequency **f t** [**Hz**].
- The delay δ t programmed by the counter c is $\boldsymbol{\delta t = tC = \frac{C}{f}}$.
- The motor step angle α , position θ , and speed ω are given by :

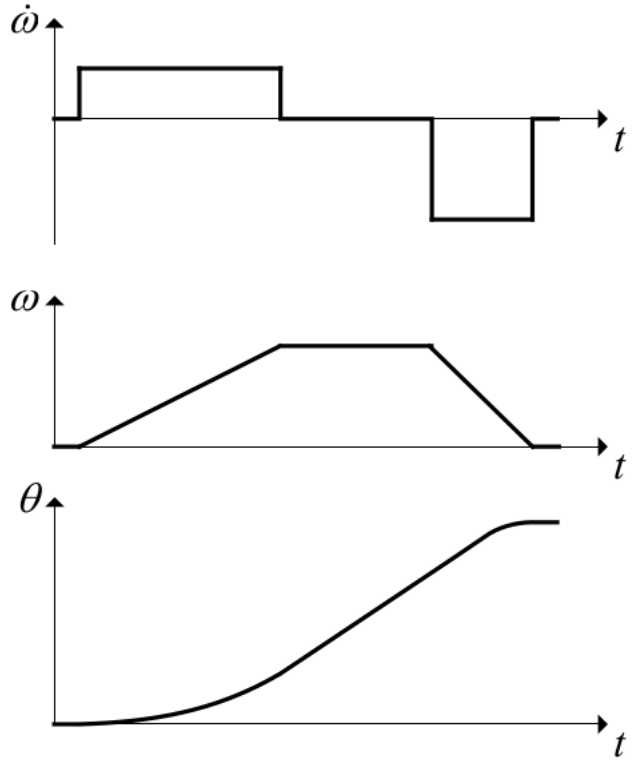
$$\boldsymbol{\alpha = \frac{2\pi}{spr}[rad], \quad \theta = n\alpha[rad], \quad \omega = \frac{\alpha}{\delta t}[rd/s]}$$

where **spr** is the number of steps per round, **n** is the number of steps, and 1 rad/sec = 9,55 rpm.

4.2 **Linear speed ramp**

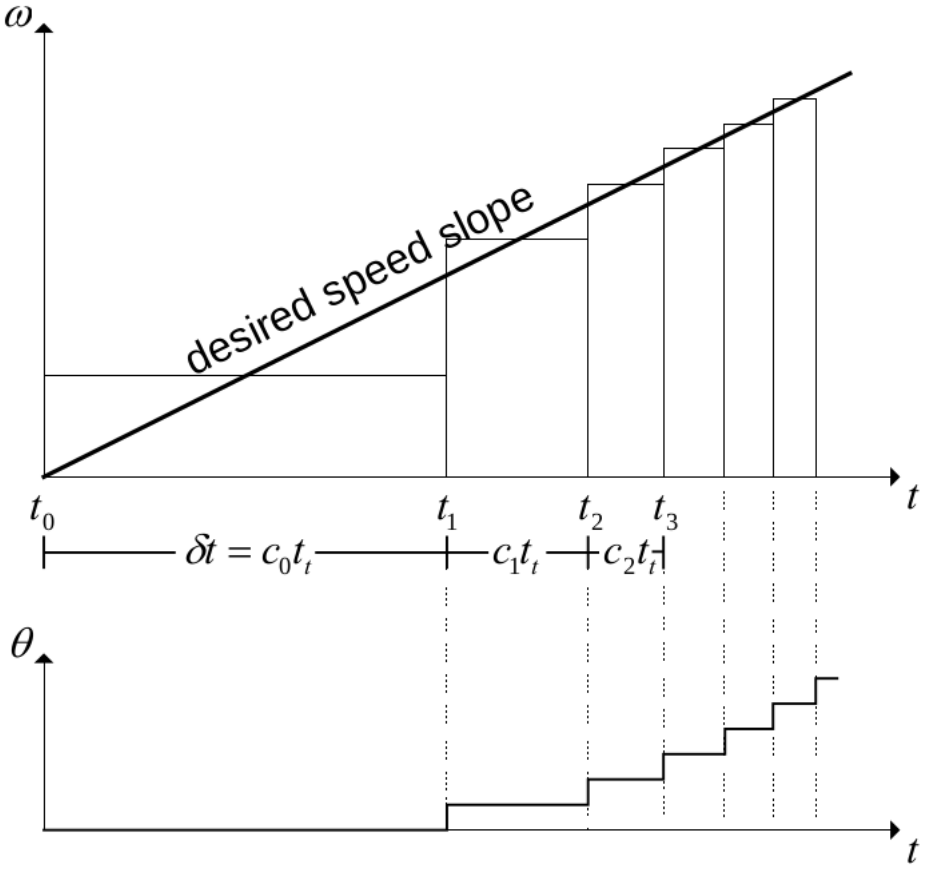
To start and stop the stepper motor in a smooth way, control of the acceleration and deceleration is needed. Figure shows the relation between acceleration, speed and position. Using a constant acceleration/deceleration gives a linear speed profile.

Figure: Acceleration (ω &) , speed (ω) and position (θ).



- The time delay δt between the stepper motor pulses controls the speed.
- These time delays t must be calculated in order to make the speed of the stepper motor follow the speed ramp as closely as possible.
- Discrete steps control the stepper motor motion, and the resolution of the time delay between these steps is given by the frequency of the timer.

Figure. Speed profile vs. stepper motor pulses/speed.



4.3 **Exact calculations of the inter-step delay**

The first counter delay c_0 as well as succeeding counter delays c n , are given by (see appendix for details):

$$\boldsymbol{C_0 = \frac{1}{t_t}\sqrt{\frac{2\alpha}{\dot{\omega}}}, \quad C_n = C_0(\sqrt{n+1} - \sqrt{n}}$$

The counter value at the time n , using Taylor series approximation for the inter-step delay (see appendix for details) is given by:

$$\boldsymbol{C_n = C_{n-1} - \frac{2C_{n-1}}{4n+1}}$$

- The acceleration is given by c 0 and n . If a change in acceleration (or deceleration) is done, a new n must be calculated.
- The time t n and n as a function of the motor acceleration, speed and step angle are given by

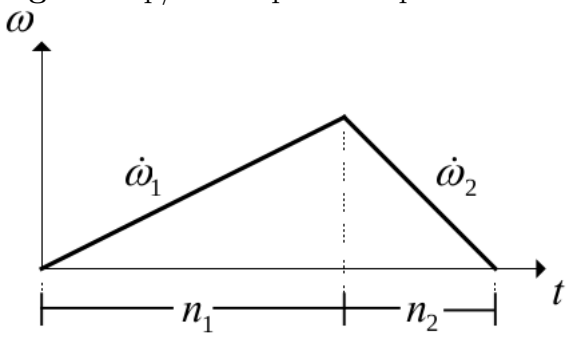
$$\boldsymbol{t_n = \frac{\omega_n}{\dot{\omega}}}$$

- Merging these equation gives the relationship

$$\boldsymbol{n\dot{\omega} = \frac{\omega^2}{2\alpha}}$$

- This shows that the number of steps needed to reach a given speed is inversely proportional to the acceleration: $\boldsymbol{n_1\dot{\omega}_1 = n_2\dot{\omega}_2}$

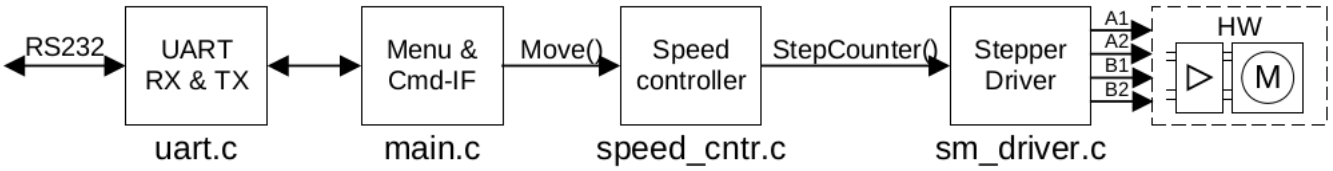
Figure. Up/down speed ramp



5. **Implementation**

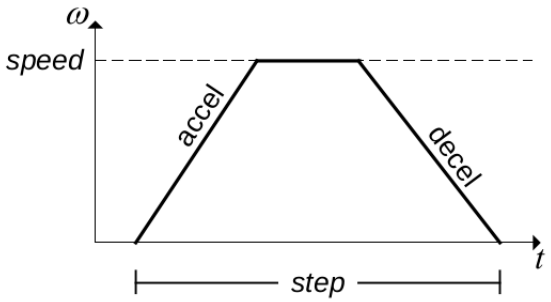
The demo application is divided in three major blocks, as shown in the block diagram in Figure. There is one file for each block and also a file for UART routines used by the main routine.

Figure 3-1. Block diagram of demo application



- main.c has a menu and a command interface, giving the user control of the stepper motor by a terminal connected to the serial line.
- speed_cntr.c calculates the needed data and generates step pulses to make the stepper motor follow the desired speed profile.
- sm_driver.c counts the steps and outputs the correct signals to control the stepper motor.

Figure Speed profile



The parameters describing the speed profile is:

- step - Number of steps to move.
- accel - Acceleration to use.
- decel - Deceleration to use.
- speed - (Maximum) speed to use.

3.5.1 Menu and command interface

The UART setting is 19200 baud, 8 data bit, none parity and 1 stop bits. Any terminal emulation program should work. Using the terminal the user can give different commands to control the stepper motor and get information back from the demo application.

The UART RX interrupt routine (found in uart.c) stores received characters in the receiver buffer and handles backspace. When ¡enter¿ (ascii code 13) is received the main routine reads the buffer and executes the given command.

- When starting, and on the '?' command, this help screen is shown:

```
-------------------------------------------------------------
Atmel AVR446 - Linear speed control of stepper motor

?        - Show help
a [data] - Set acceleration (range: 71 - 32000)
d [data] - Set deceleration (range: 71 - 32000)
s [data] - Set speed (range: 12 - motor limit)
m [data] - Move [data] steps (range: -64000 - 64000)
move [steps] [accel] [decel] [speed]
         - Move with all parameters given
<enter>  - Repeat last move

    acc/dec data given in 0.01*rad/sec^2 (100 = 1 rad/sec^2)
    speed data given in 0.01*rad/sec (100 = 1 rad/sec)
-------------------------------------------------------------
```

- After the menu is shown or a command is executed the info line is shown:

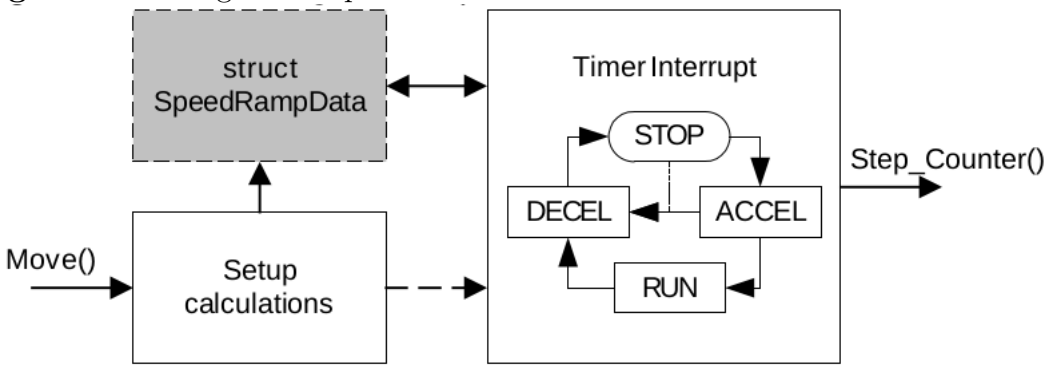
  Motor pos: 0 , a:4000, d:4000, s:2000, m:400

- The demo application gives the current motor position, acceleration, deceleration, and speed settings, as well as the number of steps to move.

3.5.2 Speed controller

The speed controller calculates and generates the speed profile.To run the stepper motor, the speed controller is set up by calling the function Move().

Figure Block diagram of speed controller

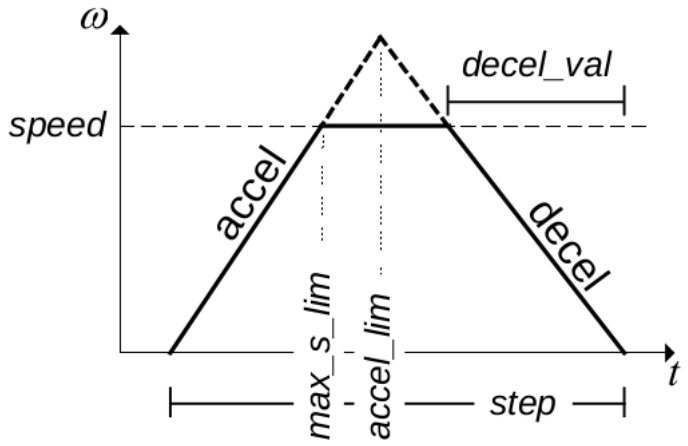


3.5.3 Setup calculations

- Find speed:
  - A_T_x100 = αf_t*100
  - min_delay=C=$\frac{A\_T\_x100}{speed}$
- Find acceleration:
  - T1_ FREQ_148 =0.676f_t/ 100

- A_SQ = $2\alpha \times 10000000000$
- step_delay=$C_0 = T1\_FREQ\_148\sqrt{\frac{A\_SQ}{accel}}/100$

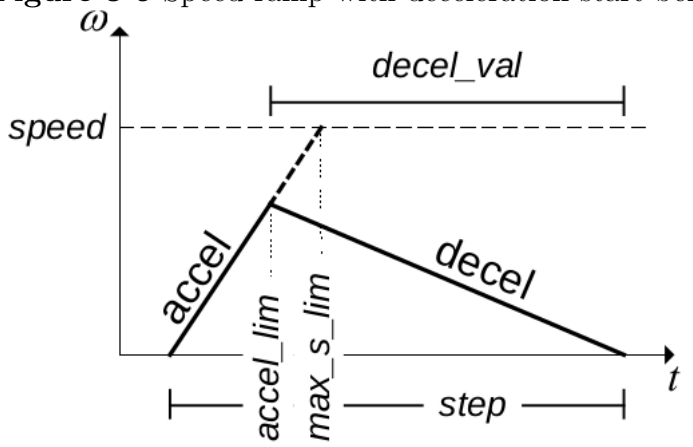
- Acceleration continues until desired speed is reached
  Figure Speed ramp limited by desired speed value.



- max_s_lim is the number of steps needed to accelerate to the desired speed. max_s_lim=n=$\frac{speed^2}{2\alpha * accel * 100}$
- accel_lim is the number of steps before deceleration starts (disregarding desired speed).
  accel_lim=$n_1 = \frac{step * decel}{accel + decel}$

  Ifmax_s_lim ¡ accel_lim the acceleration is limited by reaching desired speed. The deceleration depends on this, and in this case decal_val is found by: decel_val=-max_s_lim*$\frac{accel}{decel}$

- Deceleration starts before desired speed is reached
  Figure 3-5 Speed ramp with deceleration start before desired speed reached



If max_s_lim ¿ accel_lim the acceleration is limited by deceleration start, decal_val is then found by:
$decal_val = -(step - accel_lim)$

3.5.4 Timer interrupt

The timer interrupt generates the 'step pulse'(calls the function StepCounter()) and is only runnig when the stepper motor is moving.The timer interrupt will operate in four different states according to the speed profile,
Figure3-6: Operating state for different speed profile parts

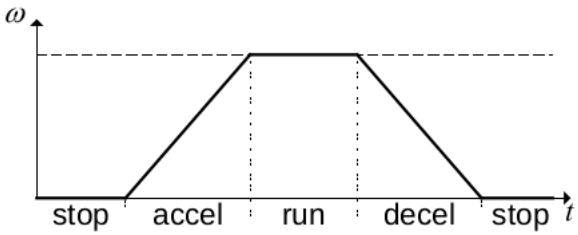
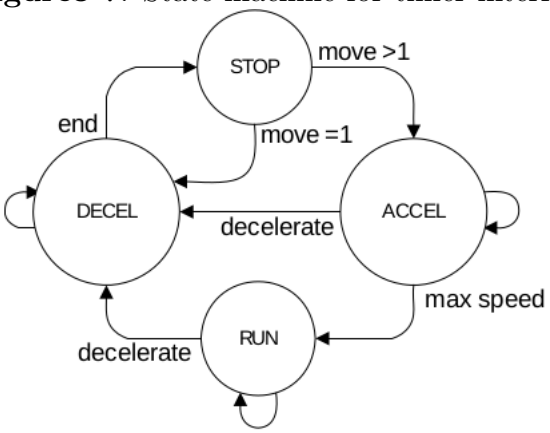


Figure3-7: State machine for timer interrupt



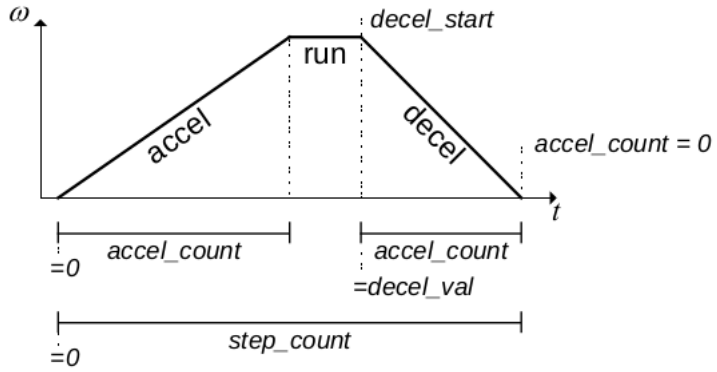
Calculations and counters

For each step during acceleration and deceleration a new time delay must be calculated. This calculation includes a division giving a remainder, and to improve accuracy this remainder is kept and included in the next calculation.

$$\boldsymbol{new\_step\_delay = step\_delay - \frac{2*step\_delay+rest}{4*accel\_count+1}}$$

$$\boldsymbol{new_rest = (2 * step\_delay + rest)(mod(4 * accel\_count + 1))}$$

Figure3-8: Counting variables in time delay



6. **Code size and speed**

   The timer interrupt performs calculations during acceleration and deceleration and approximately 200us are used in one timer interrupt. When running at constant speed less time is need and approximately 35us is sufficient. The maximum output speed is then limited by acceleration/deceleration calculations. For a stepper motor with 400 steps per round maximum output speed is:

$$\boldsymbol{\omega_{max} = \frac{2\pi}{200us * 400} = 78,5 rad/sec (= 750 rpm)}$$

   where Rpm=(rad/s)*$(2/\pi)$

7. **How to connection Steper_Motor with TB6600**

8. **How to Configure in STM32**

   - **Step1:** Click TIMER4 for Internal Clock (select TIM4 $->$ click Internal clock and set (Prescalar(89) and Autoreload(9999)) $->$ click NVIC go to click global Interrupt .
   - **Step2:**Create function Microsecond delay (Select TIM9 $->$ click Internal Clock $->$ set(Prescaler(179), Autoreload(0xffff-1).
   - **Step3:** Select PIN Output two for Control Step and Direction

9. **Setup code in main.c**

   9.1 **Control stepper not has Library**

   - **Step1:**Create function Microsecond delay.
   - **Step2:**Create function Control Stepper Motor

```c
void microDelay (uint16_t delay)
{

  __HAL_TIM_SET_COUNTER(&htim9, 0);
  while (__HAL_TIM_GET_COUNTER(&htim9) < delay);
  //HAL_TIM_Base_Stop_IT(&htim9);
}
void stepper_Motor (int steps, uint8_t direction, uint16_t delay)
{
  int x;
  if (direction == 0)
  {
    HAL_GPIO_WritePin(GPIOA, GPIOA_PIN_7, GPIO_PIN_SET);
  }
  else
  {
    HAL_GPIO_WritePin(DIR_PORT, DIR_PIN, GPIO_PIN_RESET);
  }
  for(x=0; x<steps; x=x+1)
  {
    HAL_GPIO_WritePin(STEP_PORT, STEP_PIN, GPIO_PIN_SET);
    microDelay(delay);
    HAL_GPIO_WritePin(STEP_PORT, STEP_PIN, GPIO_PIN_RESET);
    microDelay(delay);
  }
}
```

- **Step3:** Input function control stepper in While.
  stepper_Motor (200, 0, 500;)
  HAL_Delay(1000);

9.2 Control Stepper has Library

- **Step1:** Include library below:
  Accel_stepper.c from **Mr.Phany**.
- **Step2:**Input PIN to control stepper in library
  Accel_Stepper_SetPin(&Stepper1, GPIOA, GPIO_PIN_6, GPIOA, GPIO_PIN_7);
  Accel_Stepper_SetTimer(&Stepper1, &htim4);
- **Step3:**Input this code to while(1)
  if(Stepper1.run_status == 0)
  {
  . Accel_Stepper_Move(&Stepper1, set_theta1, 1000, 1000, 500);
  . set_theta1 = 0;//reset steps to 0 (prevent re-run after done)
  }
- **Step4:**Input this function to timer Interrupt
  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
  {
  . if(htim-¿Instance == TIM4)
  . {//stepper1
  . Accel_Stepper_TIMIT_Handler(&Stepper1);
  . }
  }