S79N [Regular version] Parameter description (version 1)

Chapter 1 Performance Description

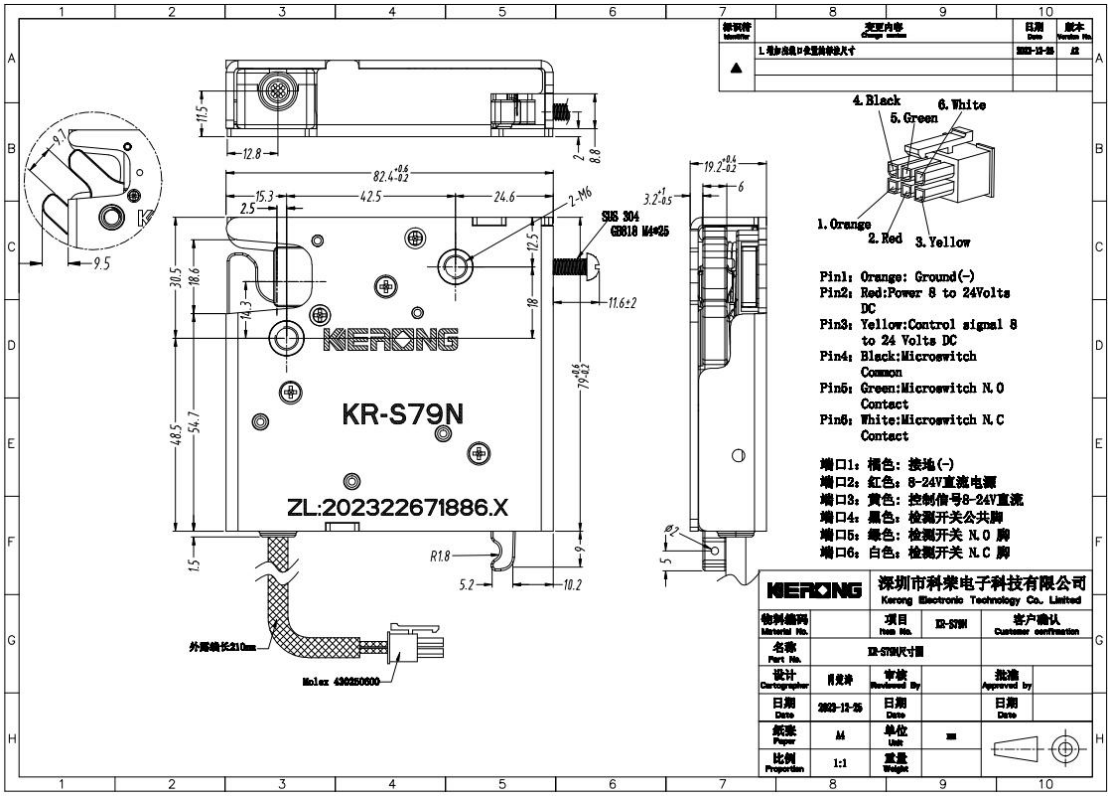
S79N(Metal motor-S79N motor)						
Working Current	power consumption	Static current (signal unlock)	RMP	Response time	Working life	Working Voltage
80mA@12V	0.96W	Less than 20uA	74+7rpm	>1000ms	>200,000 times	8V-24V Typical 12V
Recommended user-powered unlocking time greater than 1000ms (uninterrupted)						

Product	Information
Picture	
Model	KR-S79N
Brand	KERONG
Size	79*82.4*19.2mm
Weight of lock body	420g
Weight of hook A	76g
Weight of hook B	29g
Weight of hook C	40g
Material of lock body	304 Stainless steel
Material of lock tongue	Zinc Alloy
Door weight	0.5KG~50KG
Maximum force-hold strength	500kgf
Driven method	Motor/ servo
Working voltage	DC12V and DC24V and 8-24V
Suggest powering time	more than 10000ms(uninterruptedly)
Working principle	Signal lock & unlock
Signal detection switch	Omron
Signal detection work mode	Feedback the lock status when unlocked & locked
Working temperature	-20℃ ~ +65℃
Working humidity	5% ~ 85%RH
Working life	200,000 times
IP rating	IP66
Test	Salt spray testing 72 hours
certification	CE，RoHS 2.0

Chapter 2 Outlet Port Description

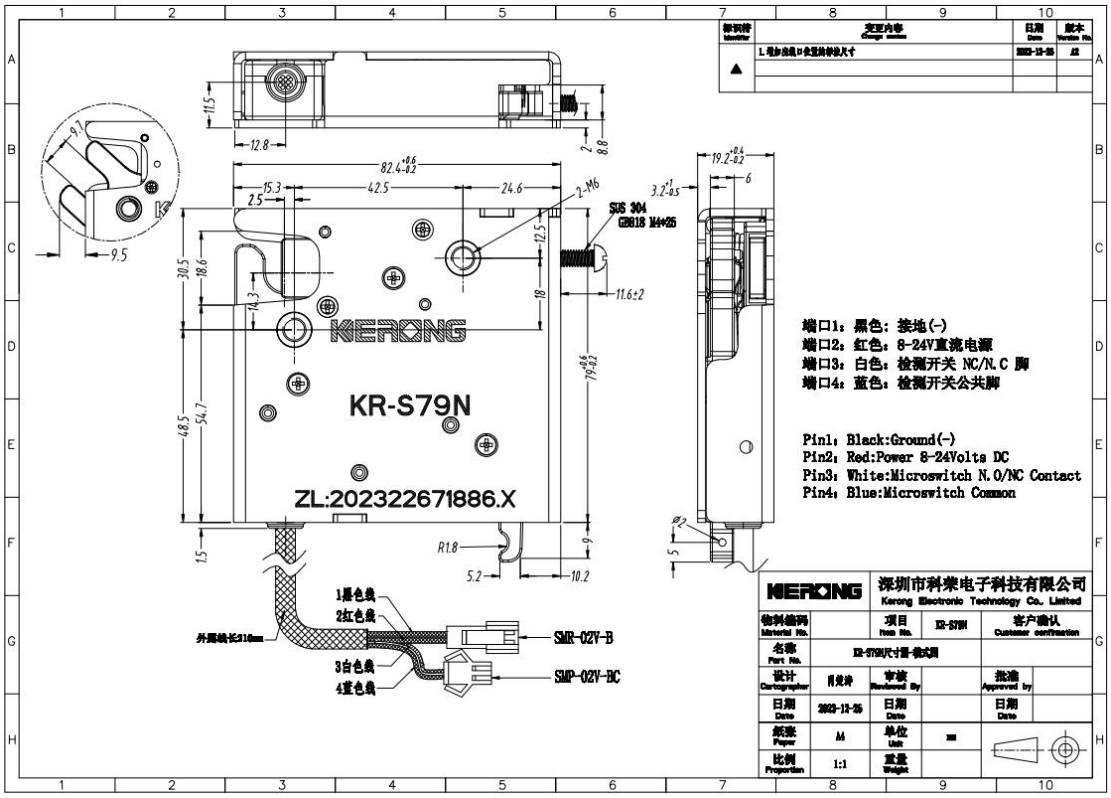
Dimension（Signal mode 1，Signal mode 2，Signal mode 3）

Connector with 6pin



Dimension（Signal mode 4）

Connector with 2*2pin



1. Connector（input pole）Orange【1】： Power supply negative (GND);
2. Connector（input pole）Red【2】： Power supply positive (8V~24V);
3. Connector（input pole）Yellow【3】： Unlock signal (3 modes optional);

Signal mode 1: When the pin [3] receives a high-level pulse signal more than 100ms, it will be unlocked or locked. This signal is a wide voltage of 5~24V; (as shown in Fig. 2)

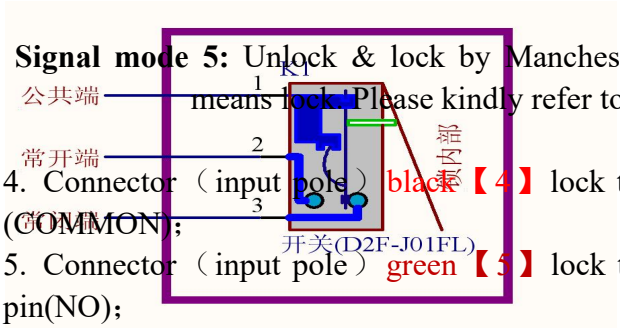
Signal mode 2: When the pin [3] receives a high level signal, the lock will be unlocked. When the pin[3] receives a low level signal, the lock will be locked. The high level signal is a wide voltage of 5~24V, and the low level signal is a voltage less than 2V; (as shown in Fig. 3)

Signal mode 3: When the pin [3] receives a high level signal, the lock will be locked. When the pin[3] receives a low level signal, the lock will be unlocked. The high level signal is a wide voltage of 5~24V, and the low level signal is a voltage less than 2V; (as shown in Fig. 3)

Notice: Under signal mode 2 and 3, if receives a continuous high level signal, lock system will not hibernate.

Signal mode 4: Unlock & lock when power on, this mode without signal cable. When power on, lock will unlock & lock for once.

Signal mode 5: Unlock & lock by Manchester code. 0X5B means unlock, 0x5F means lock. Please kindly refer to the Fig 5 and 6, and the source code.



4. Connector（input pole）black【4】lock tongue detective switch: common pin (COMMON)；
5. Connector（input pole）green【5】lock tongue detective switch: normal open pin(NO)；
6. Connector（input pole）White【6】lock tongue detective switch: normal close pin (NC)；

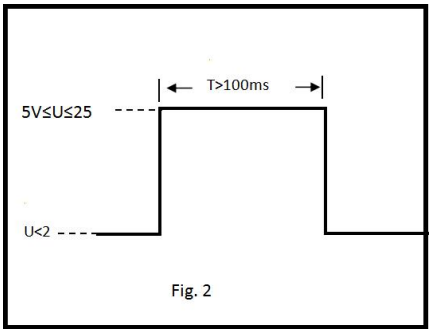


Fig. 2

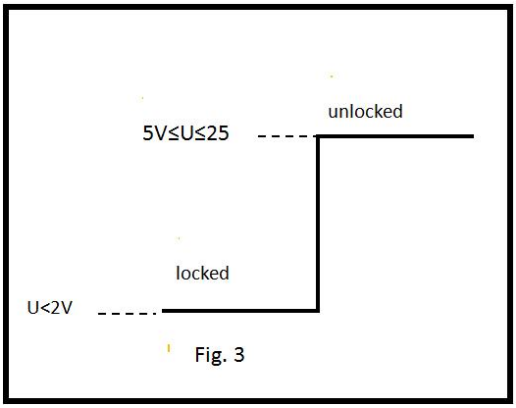


Fig. 3

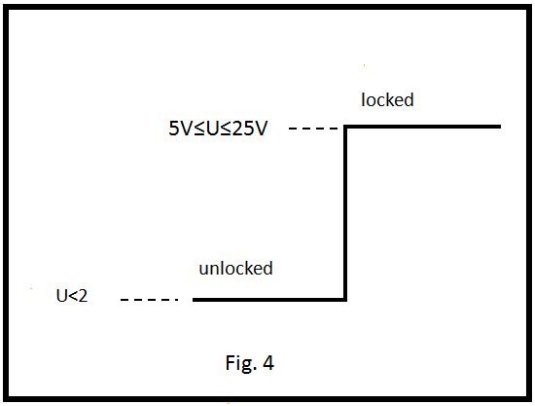


Fig. 4

Fig. 5 (0x5B)

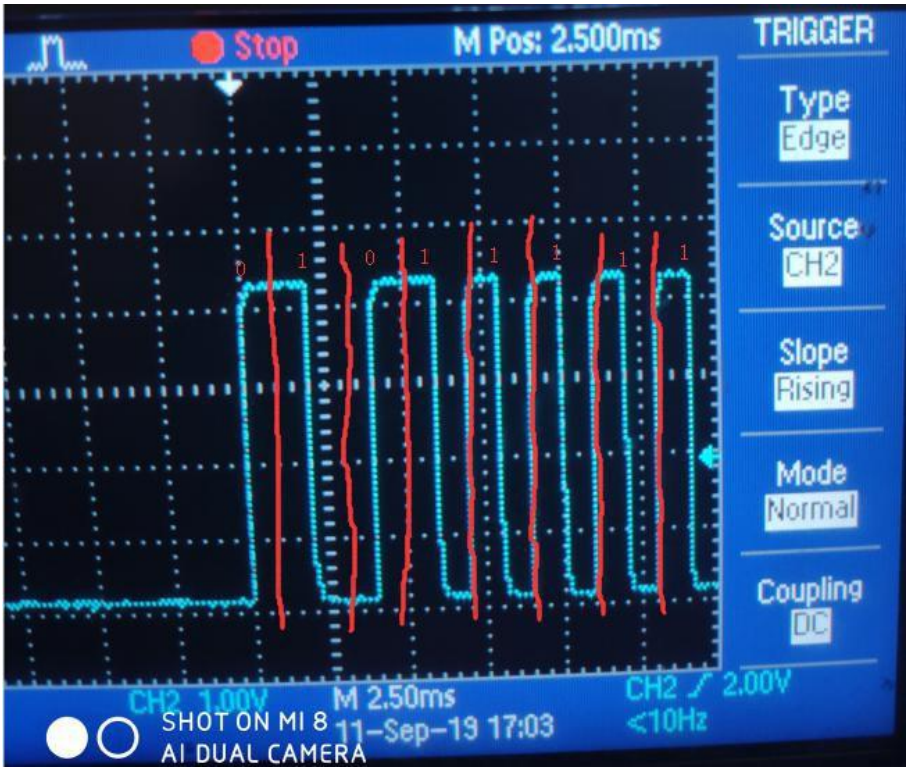
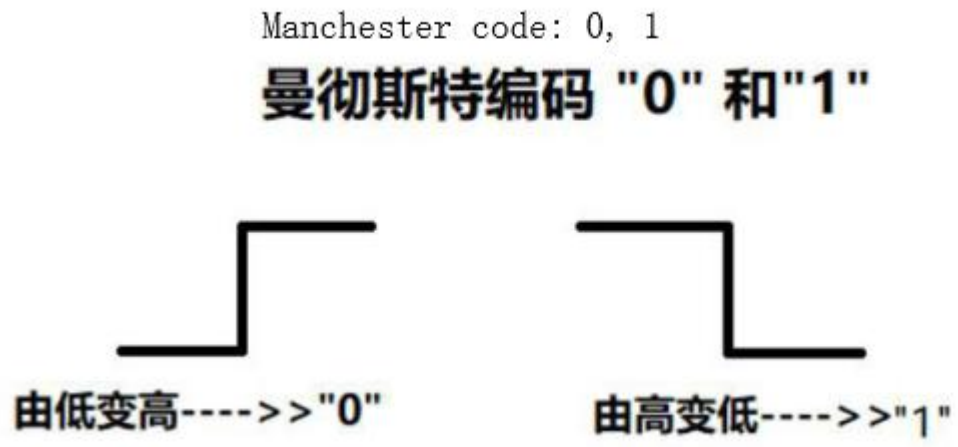


Fig. 6 (0x5F)



Notice: cycle of one data is 2ms

```
void CreatSignal(char Data)
{
char i;
if(Data == 0x5F)

for(i=0;i<4;i++)  //send for 4times repeatedly
{
gpio_out_low(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_high(OPEN_LOCK_SIGNAL);       //0
TIM2_DelayMs(1);

TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);       //1
TIM2_DelayMs(1);

TIM2_DelayMs(1);
gpio_out_high(OPEN_LOCK_SIGNAL); // 0
TIM2_DelayMs(1);

TIM2_DelayMs(1);
  gpio_out_low(OPEN_LOCK_SIGNAL);      //1
TIM2_DelayMs(1);

gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);       //1
TIM2_DelayMs(1);
```

```
gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);

gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);

gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);
}
}
if(Data == 0x5B)
{
for(i=0;i<4;i++) //send for 4times repeatedly

gpio_out_low(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_high(OPEN_LOCK_SIGNAL);       //0
TIM2_DelayMs(1);

TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);

TIM2_DelayMs(1);
gpio_out_high(OPEN_LOCK_SIGNAL); // 0
TIM2_DelayMs(1);

TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);

gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);



TIM2_DelayMs(1);
```

```
gpio_out_high(OPEN_LOCK_SIGNAL);       //0
TIM2_DelayMs(1);


TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);

gpio_out_high(OPEN_LOCK_SIGNAL);
TIM2_DelayMs(1);
gpio_out_low(OPEN_LOCK_SIGNAL);        //1
TIM2_DelayMs(1);
}
}
}
```