Theory:

To enable use of the TCA0 timer/counter for counting, first an origin statement needs to be made to connect TCA0_OVF_vect, the overflow vector for TCA0, to an interrupt service routine (ISR).
Next, in the configuration of the program, select normal mode for the timer/counter by writing TCA_SINGLE_WGMODE_gc, or group configuration for normal mode, to TCA0_SINGLE_CTRLB, or the register that controls the counter mode. This will make the counter run as a 16 bit binary counter.
Two enable overflow interrupt, set the TCA0_SINGLE_INTCTRL, or the interrupt control in single operation mode, to 0x01.
Next the period for the counter has to be set. For this, write the low byte of the desired period to TCA0_SINGLE_PER and the high byte to TCA0_SINGLE_PER + 1. The value in the period register determines when the counter will hit TOP, ie. when it will create an overflow interrupt.
Next the clock divisor needs to be set. The divisor is the bits 4 to 1 of the register TCA0_SINGLE_CTRLA. Set this to the desired divisor.
While working with TCA0_SINGLE_CTRLA, set bit 0 to 1 to start timer
Then, enable global interrupts.
Inside the ISR, first make sure to push to the stack any registers you plan on using in the ISR, as well as the status register.
Perform any desired periodic operations, and then clear the overflow interrupt flags
To clear the flags, write a 1 to bit 0 of the TCA0_INTFLAGS register
Make sure to pop off the stack any registers that were stored there, as well as restore the status register.

To sum up, the TCA0_SINGLE_CTRLB register determines the operation mode, the TCA0_SINGLE_INTCTRL register determines the interrupts to be used, the TCA0_SINGLE_PER register determines the period of the counter, and the TCA0_SINGLE_CTRLA register determines the clock divisor, as well as enables the timer.
The origin statement directs the program to the ISR whenever an interrupt occurs, and the ISR then clears the flag once it has performed the desired periodic operation.

Using the example of a simple periodic waveform, the ISR toggles an output by xoring the value off the pin with a bitmask with a 1 in the position of the output pin. The value written into the PER register should be half of the desired period, as the program toggles the pin. The CTRLA register can be used to slow the clock if needed.