![Renesas]

AN-1192 Power Duty Cycle Controller to Prolong Battery Life, with MCU Programmable Duty Cycle

Some battery powered products, particularly Internet of Things (IoT) sensor transceivers, are best powered intermittently, to prolong battery life. For example, say a transceiver only needs to check for a signal once per second. Based on this period, a duty cycle controller turns on power to the transceiver and controlling MCU. Once the MCU has determined that the transceiver has completed its work, it triggers the duty cycle controller to restart by turning off the power to the downstream circuit.

Existing Technology

This duty cycle configuration is currently achieved by using multiple connected devices, such as TI's TPS61291, TPL5111, or CD4541, as well as associated passive components.

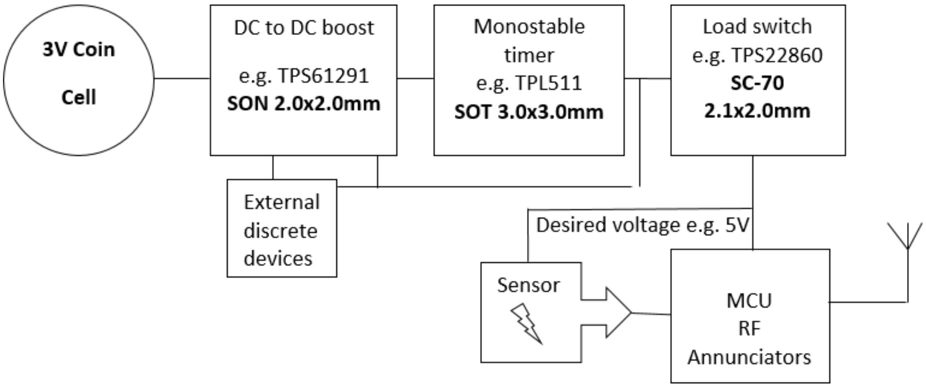

Figure 1. Example of a setup using purpose-made discrete devices to achieve power duty cycling

GreenPAK™ Technology

The ability to modify the delay period between each cycle is advantageous, as it allows the application to modify the duty cycle for the best battery life by taking into account external factors (e.g. time of day, season, temperature, traffic volumes, etc.).

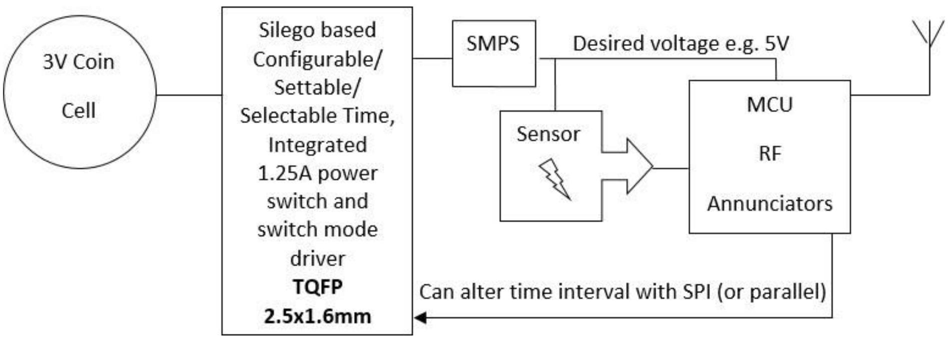

Figure 2. Example of the GreenPAK SLG46116V/SLG46117V based replacement, with a smaller footprint, fewer supporting passive components, and optionally enhanced duty cycle control with MCU modifiable interval

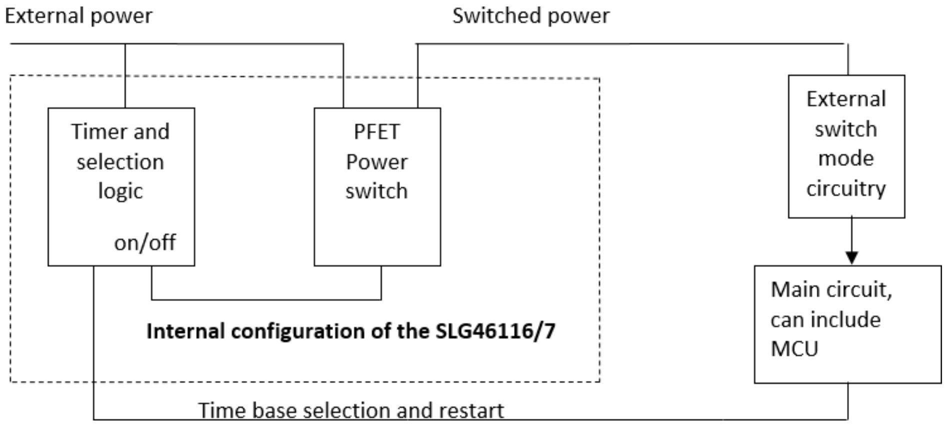

Figure 3. Block diagram of SLG internals (inside dotted line)

Implementation

The circuit is implemented using a SPI interface to set and change the time interval. When the external circuit is not powered, it is expected that the input pins would have high impedance or be held at logic LOW.

It is left to the user to implement a hardwired version if the time interval is constant or a parallel interface version if preferred.

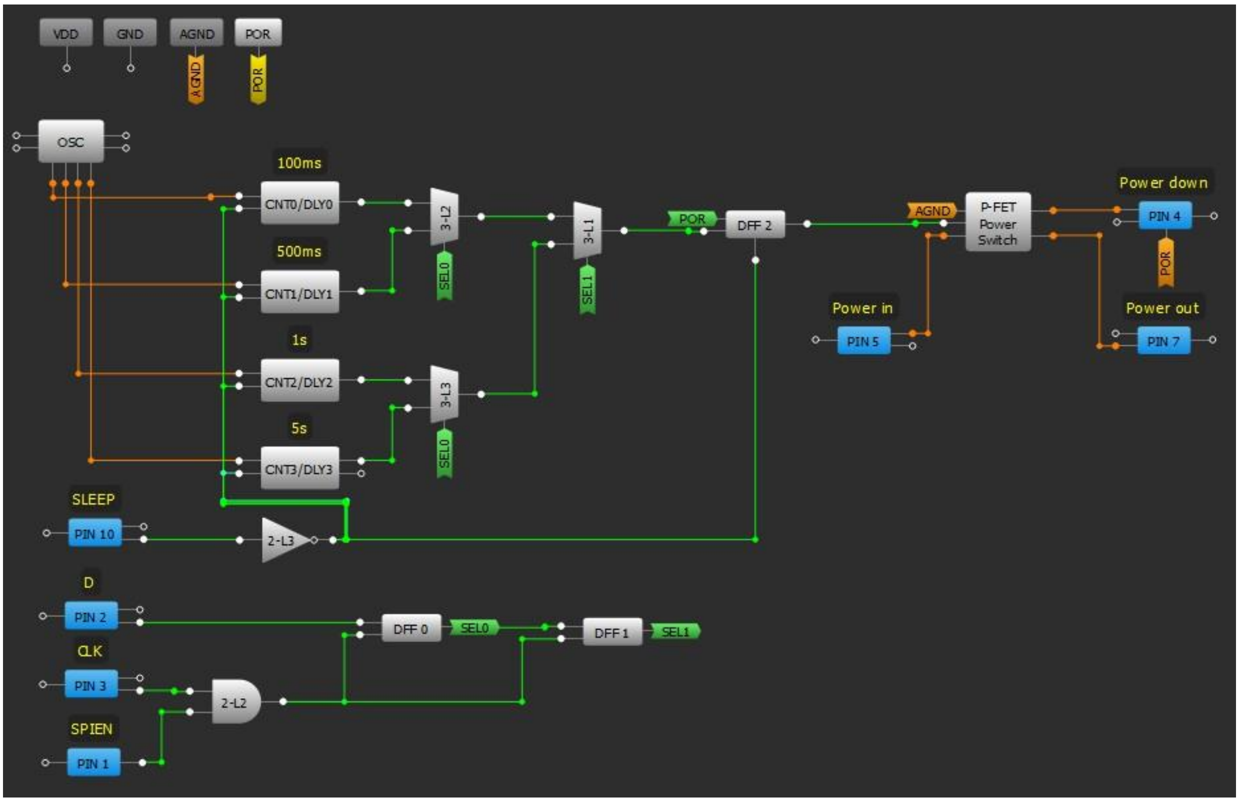

Figure 4. Power duty cycle controller GreenPAK™ schematic

Counter blocks CNT0-CNT3 are configured as rising edge delays with various delay times. When the DLY_IN signal to each of the blocks goes HIGH for 100ms, 500ms, 1s, and 5s respectively, the blocks will output HIGH. The select pins of each of the 3 muxes are connected to DFF0 and DFF1, whose operation is described below. The selected MUX output will go HIGH at the end of the delay period and will drive the P-FET Power Switch on, connecting Power in to Power out. Once the MCU or other external hardware has determined that the required operations have been completed, the Sleep pin is driven HIGH, and after inversion through 2-L3, this signal resets DFF2 and the CNT/DLYs, resulting in DFF 2's output going LOW. This turns off the P-FET switch. At this time, the Sleep input should return to LOW or voltage free (the pin is internally pulled down) as the external circuit loses power.

To program the selection of the time period, with SPIEN driven HIGH, the logic level at pin D is clocked into DFF0 on the rising edge of CLK. Data already in DFF0 is clocked into DFF1. This sets upSEL0 and SEL1. As most MCUs have a minimum SPI data frame size larger than two bits, it should be noted that only the last two clocked bits are retained (so you should only set up the last two bits transmitted when implementing the SPI interface on the MCU). SPIEN should remain LOW or at high impedance (it is internally pulled LOW) when the SPI interface is not in use.

At first power up, DFF0 and DFF1 have default values of 0, which results in the shortest delay. Additionally, DFF2 has an initial polarity of HIGH, turning on the P-FET switch. At this time, the desired delay should be programmed. Afterwards, the programmed delay interval will be retained until a power loss to the GreenPAK chip, or until the delay is reprogrammed.

The shown delay times can be modified to any allowable delay period by reconfiguring the Counter data (and/or Clock) for each CNT/DLY module and/or the OSC module.

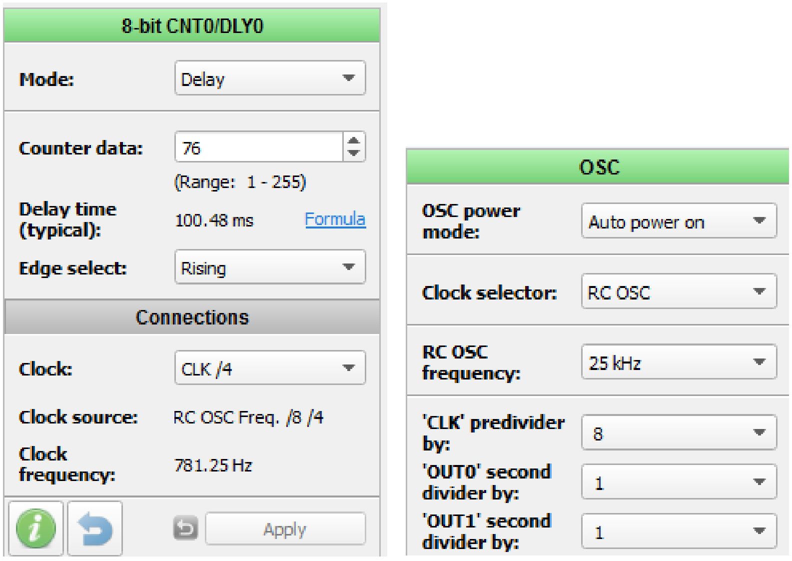

Figure 5. Predefined delays can be changed by modifying the configuration

If the SPIEN function is not required, the 2-L2 gate can be removed, and the CLK can drive DFF0 and DFF1 directly. In this case, CLK should be held LOW or be at high impedance when not clocking data.

Last 2 bits L to R	SEL0	SEL1	Selected	Delay (ms)
00	0	0	CNT0/DL0, CNT2/DLY2, 3L-2	100
01	1	0	CNT1/DLY1, CNT3/DLY3, 3L-2	500
10	0	1	CNT0/DL0, CNT2/DLY2, 3L-3	1000
11	1	1	CNT1/DLY1, CNT3/DLY3, 3L-3	5000

Table 1. Programmed bits, intermediate selections, and results

The SPI input has been tested at a clock speed of 8MHz, and works in SPI mode 0, 1, and 3.

Waveforms

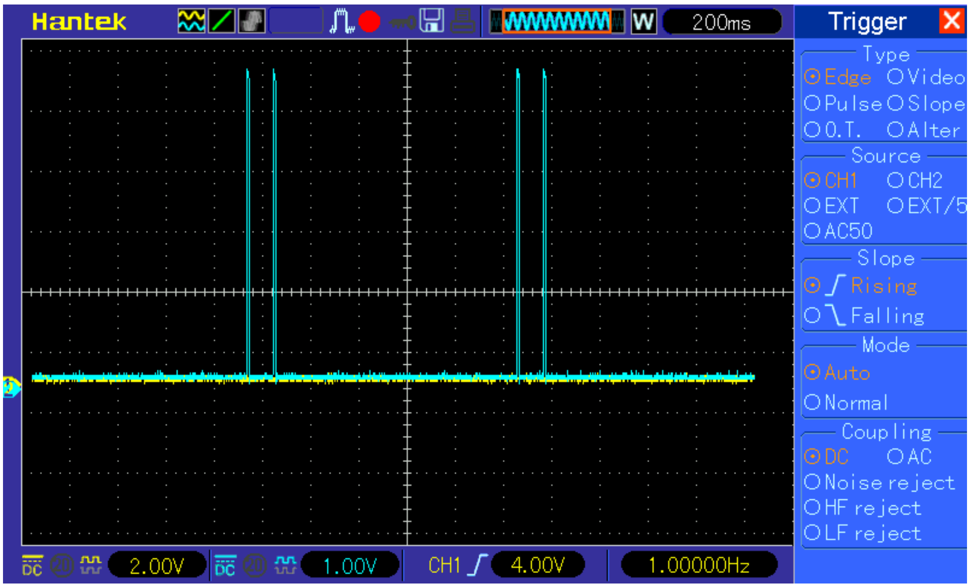

Figure 6. Demonstration of changing delay from 100 ms to 1 s

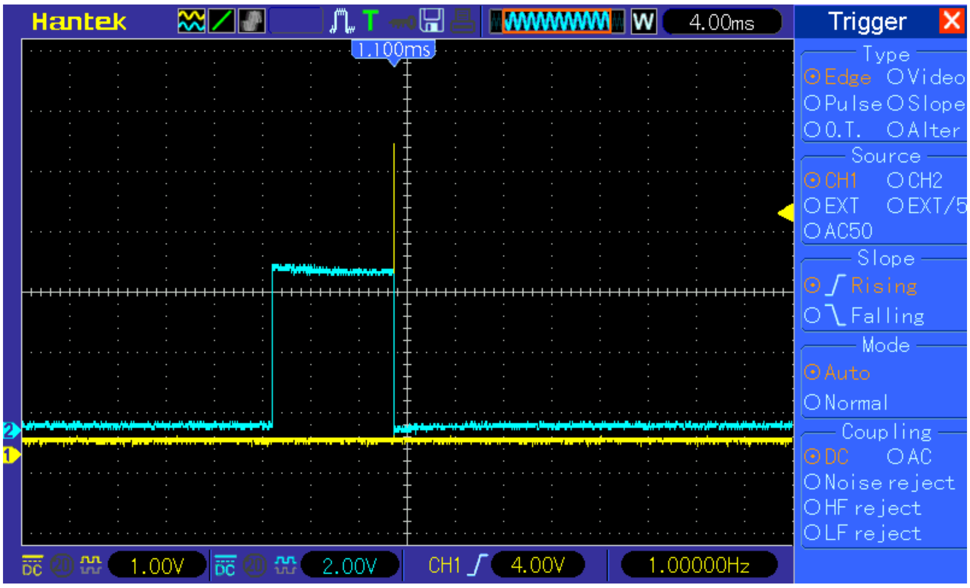

Figure 7. Showing the sleep pulse (yellow) to end the cycle

Arduino Code

These tests and demonstrations were completed using the following Arduino test code:

```
/*  Code to test duty cycle controller SPI version
  Sends 0b00 and 0b10 alternatively
  created 02 March 2017
*/

// include the library:
#include <SPI.h>
byte writeb010 = 0b010; //
// pins used for the connection with SLG
// SPI pins are controlled by the SPI library):
const int spienPin = 7;
const int sleepPin = 6;
const int powerDownPin = 5;

uint8_t prevCondition = 0;
void setup() {
  // start the SPI library:
  SPI.begin();
  // initialize the  spien pins - set level before direction to prevent
false triggering
  digitalWrite(spienPin, LOW); //do not enable comms
  pinMode(spienPin, OUTPUT);
  digitalWrite(sleepPin, LOW); //do not restart
  pinMode(sleepPin, OUTPUT);
  pinMode(powerDownPin, INPUT);
}

void loop() {
  while (!digitalRead(powerDownPin));//wait for power on - required for
emulation only
  delay(10);//do some work
  prevCondition = 1 - prevCondition;// to change condition
  writeb010 = prevCondition ? 0b10 : 0b00; //1s:100ms
  //change time period
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE1));
  digitalWrite(spienPin, HIGH); // enable comms
  SPI.transfer(writeb010);
  digitalWrite(spienPin, LOW); //stop comms
  digitalWrite(sleepPin, HIGH); //work done so restart SLG
  digitalWrite(sleepPin, LOW); //power off - required for emulation
}
```

Function	SLG46117V pin	Arduino pin
SPI Data	2	11
SPI Clock	3	13
SPI Enable	1	7
Power Down	4	5
Sleep	10	6

Table 2. SLG46117V Arduino interconnections used in demonstration

Conclusion

In this app note we created a variable-length power duty cycle controller to help prolong the battery life of a system. It allows a microcontroller to decide how long to power down and when to wake back up. Since GreenPAK IC's quiescent current is less than a typical microcontroller's quiescent current, using the GreenPAK as a dedicated wake/sleep device will help the system operate for a longer period of time before the battery runs out.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.