Memo to: Mr. Cory Mettler
From: Julian Rechsteiner
Date: February 9, 2021
Regarding: EELE 465, Lab 2 – Real Time Clock and I2C

Summary:

The purpose of this lab was to learn about "bit-banging" I2C using an MSP430-FR2355 microprocessor. This was accomplished in three large steps. The first task was to construct a basic structure of the I2C protocol using solely the microprocessor, meaning that the processor acted as a master and a slave simultaneously. The second task was to interface with a real time clock (RTC) using the I2C protocol to retrieve the current seconds, minutes, and hours. Finally, the third step required the RTC to send the same data as the previous step along with the current temperature of the RTC.

Setup:

This project did not require an extensive amount of setup; the logic channels 1 and 2 from the Analog Discovery 2 (AD2) were connected to ports 1.0 and 6.6 of the MSP430, respectively, along with its ground. For the RTC portion of the experiment, the RTC had two connections to the microprocessor (Vcc and ground), and a breadboard was used to connect the Shared Clock Line (SCL) and the Shared Data Line (SDL) to both the MSP430 and the AD2. Fig 1 depicts the circuit diagram of this experiment.

First Step – Send Data using the I2C protocol (Part 3):

The first large task of this experiment was to send dynamically generated data using the I2C protocol. The generated data consisted of a counter from 0 to 9, where each number was sent as a payload of one byte. To correctly send the data through the I2C protocol, a start bit had to be generated followed by an arbitrary address where that data would (in theory) get sent to. This arbitrary address was 0x07. To simulate the situation where that address corresponds to a slave address somewhere on the line, an acknowledge bit (ACK) was sent again from the master. Once the address and the ACK were sent successfully, the counter data from 0 to 9 were sent as a payload of one byte. These payloads were separated by an ACK bit that was, again, generated by the master for the purpose of this experiment. When the numbers 0 through 9 were successfully sent, a non-acknowledge bit (NACK), otherwise known as a logic high, was sent to complete this protocol, followed by a stop bit (sending a logic high when the clock is high). Fig 2 depicts this process in greater detail.

Second Step – Interface with an RTC to get the current RTC time (Part 4):

The second task of this experiment, Fig 3, consisted of interfacing with an RTC to retrieve its current seconds, minutes, and hours. The steps to achieve this were similar to the previous step. A start bit had to be sent to kickstart the I2C protocol, followed by the address of the slave (0x68 for the RTC) and the register address of where the master device wanted to read from. In this particular case, the seconds, minutes, and hours registers were located in the registers 0x00, 0x01, and 0x02 respectively, so sending 0x00 was sufficient as the RTC possessed an autoincrement feature for its registers. For the ACK bits, instead of the master sending the ACK, the subroutine changed to the master waiting for the ACK bit. This was performed by changing the SDL from an output to an input for a short period of time. When the register address was successfully sent, the master ended and restarted the I2C protocol. This was followed by establishing a connection to the slave device once again and turning the SDL to an input in order to retrieve the necessary data from the RTC. This was implemented by reading the following 3 bytes of data and then terminating the connection.

Third Step – Interface with an RTC to get the current time and temperature (Part 5):

The third and final task of this experiment was to also retrieve the temperature from the RTC. This was done using a similar approach to retrieving the time data from the previous step. Once the MSP430 terminated the connection, it reinitialized it by sending a start bit, the address of the slave device (0x68), waited for the ACK from the slave, and subsequently sent the register address of where the temperature data resided (address 0x11). Similarly, the master devices awaited two payloads of one byte each to retrieve the temperature data from address 0x11 and 0x12.

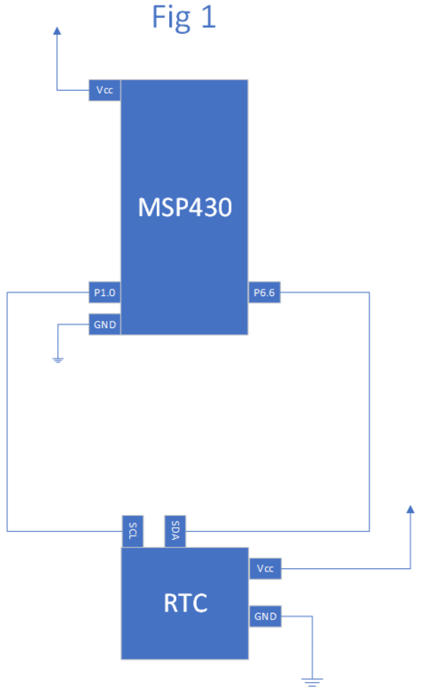
Conclusion:

The I2C protocol required several key steps to produce. A start bit is needed to initialize the protocol, followed by the slave address, an ACK bit from the slave, and the register address to notify what register the slave will send. The RTC device stored the time data in registers 0x00, 0x01, and 0x02, as well as temperature data in registers 0x11 and 0x12, so constructing the I2C protocol correctly to retrieve the data in these registers was imperative.

Lessons Learned:

- RTC only turns off when no power is connected, not when the debugger restarts.
- Use a breadboard to interface with more than two connection simultaneously
- Spend more time working on the project in the earlier part of the lab.

Fig 1



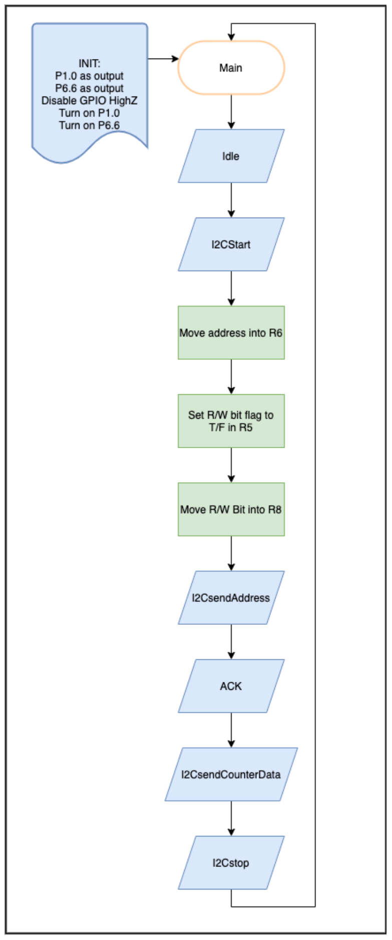


Fig 2

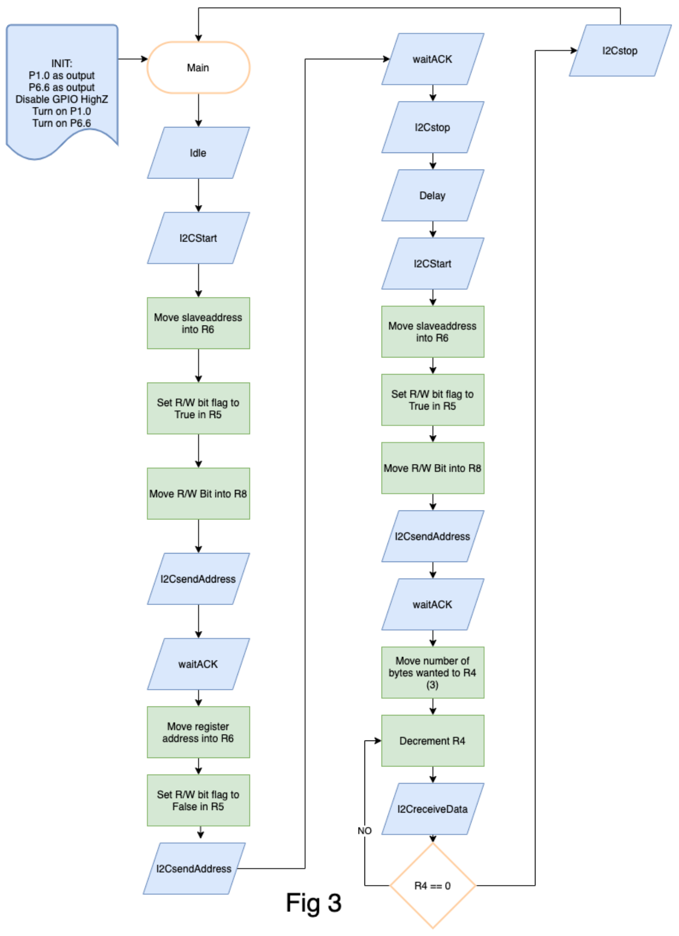


Fig 3