Chapter 1

MOTMaster

1.1 Chapter Overview

The aim of this chapter is to provide a description of the MOTMaster software, which was developed from a pre-existing version during my PhD. The design of MOTMaster assumes very little about the particular experiment it is being used for, so much of the discussion in this chapter will be kept general. This chapter begins by motivating the need to extend MOTMaster by developing a graphical interface to simplify the creation of experimental sequences, as well as implementing new methods of controlling hardware. This is followed by a general description of how MOTMaster configures the various types of inputs and outputs used in an experiment.

1.2 Motivation

In the initial stages of my PhD, I decided to use Cicero Word Generator [1] to control the hardware for the experiment. This is a graphical-based control system developed by

Wolfgang Ketterle's group at MIT, which was designed for controlling atomic physics experiments using National Instruments hardware. Over time, as the experiment became more complex, it started to become apparent that Cicero was not suited to meet all of our requirements for control software. This was most evident in the control of the M-Squared Raman laser system. Unlike the Muquans laser system, which can be controlled externally using analogue and digital voltages and serial messages, the M-Squared system is externally controlled by communicating JSON messages to a web server. Implementing such a drastically different scheme for controlling a specific component into Cicero was not deemed worthwhile. Around this time, I also realised that Cicero takes an appreciable amount of time (around 300 ms) to re-calculate the experiment sequence between each shot. Since the design of Cicero was aimed at controlling experiments that take many seconds per cycle, this dead time between each cycle is not significant on those time scales. In contrast, each cycle of this experiment takes around 250 ms. This unnecessary dead time needed to be addressed if we hoped to improve the repetition rate.

After it became clear that a potentially large amount of work would be needed to improve Cicero, I decided that it was worth moving to a new control system. A collection of programs, named EDMSuite, has been developed by people in **ccm!** (**ccm!**) to control a range of experiments within the group. One application, MOTMaster, was designed to control and acquire data from experiments investigating cold atoms trapped in a **mot!** (**mot!**), as its name suggests. For this reason, it seemed the most appropriate software for our purposes. However, its method of structuring experimental sequences was inconvenient, as it lacked a graphical user interface to do so. During the process of switching to using MOTMaster to control the experiment, I designed a graphical method of structuring sequences, which functioned identically on a device level to the original method of defining sequences. In addition to this, I included an interface to the M Squared laser system, so that it could be controlled

using MOTMaster. As with other hardware interfaces, the M Squared interface is loosely coupled to the rest of MOTMaster, so that other experiments which do not use this laser can still use MOTMaster.

1.3 Interfacing wth Hardware

The majority of the experimental hardware is controlled using analogue and digital voltages that are generated by **daq!** (**daq!**) cards manufactured by National Instruments. MOTMaster is compatible with cards that use either the NI-DAQmx or NI-HSDIO device drivers. These are used to configure the generation or acquisition of digital or analogue voltage waveforms and are capable of precisely timing and synchronising their I/O across multiple devices. Most components in the experiment rely on this precise timing to function correctly. Other devices, where timing accuracy is less critical, are controlled by sending or receiving data using serial communication. This has the advantage of allowing more structured command beyond analogue or digital voltages, but the communication speed of the serial channel limits the accuracy of the execution time. An understanding of the low-level interface between control software and the experiment is very useful in both carrying out experiments and accurately interpreting the results.

1.3.1 Hardware Abstraction

When designing software, it is often useful to structure a program in such a way that modules which make use of other components do not need to know about their specific implementation in order to use them. This approach, known as loose coupling, means that the submodule can be modified without harming the compatibility of these two components. In the context of experimental hardware, this is equivalent to requiring

that changing specific components, for example the **vco!** (**vco!**) that generates the RF power for an **aom!** (**aom!**), will not stop the experiment from working. This is done using abstract representations of the hardware, in the form of input and output channels that are used to communicate to each device.

1.3.2 Voltage Pattern Generation

All the analogue outputs controlled using MOTMaster are done using the NI-DAQmx software. Each output uses a **dac!** (**dac!**) to convert a floating-point number into an analogue voltage. To generate a sequence of voltages across multiple channels, the NI-DAQmx driver allocates a block of memory on the **daq!** card for each output channel. This memory acts as a first-in first-out (FIFO) buffer for data streamed to it from a computer. The output of each channel is synchronised to a clock signal, so that every time a rising edge occurs on the clock, the voltage at each output transitions to the value corresponding to the next value in its corresponding buffer. Channels across multiple **daq!** cards can be synchronised by sharing a clock signal, which can be done using the bus that connects cards in a PXI-e chassis. Additional cards can also be configured to trigger the start of their output at the moment they receive the first clock pulse, rather than waiting for a software trigger from the computer.

Digital outputs from NI-DAQmx cards are generated in much the same way as analogue voltages, except for the fact that they only take two values corresponding to either a low (0 V) or high (3.3/5 V) level. **daq!** cards which operate using the NI-HSDIO driver function differently. These cards have faster on-board clocks than is usually available with NI-DAQmx hardware. For instance, NI-DAQmx PXI-6723 can operate with a maximum clock frequency of 200 kHz and typical sequence durations mean that a sample rate of 100 kHz is needed to fit the entire sequence into memory. However, the NI-HSDIO PXI-6541 card can generate digital voltages at sample rates

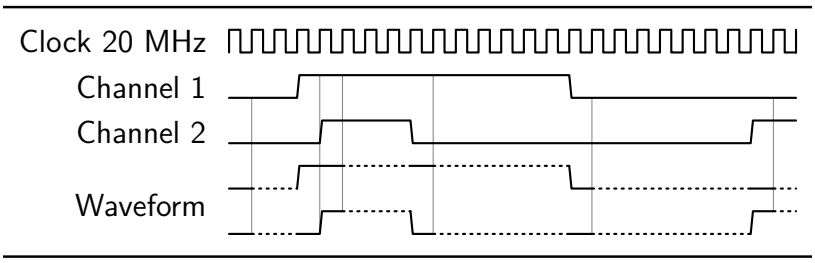

Figure 1.1: Scripted pattern generation for an NI-HSDIO digital output card. A pattern is split into segments which correspond to a duration for which all the channels output a constant value. Each of these smaller waveforms are written to the on-board memory, along with a script that instructs the card to output each pattern for the required number of times to reconstruct the original sequence. By reducing the amount of memory required to define the sequence, a faster clock frequency and hence timing resolution can be used to output digital control signals.

up to 50 MHz and requires less memory to store a pattern. Rather than write the pattern as an array of values for each sample, the sequence is segmented into smaller patterns during which the state of each channel is constant, as illustrated in Figure 1.1. NI-HSDIO cards can be scripted to generate each of these patterns for the appropriate number of clock cycles.

1.3.3 Timed Serial Communication

Serial communication is used to control devices which require more complex control than is possible using analogue or digital voltages. This increase in complexity comes at the cost of slower response times, because it takes longer to communicate an array of bytes is longer than to change the voltage across an output terminal. Using the NI-VISA driver, the output of serial data can only be timed using a software clock on a computer, which is more prone to jitter than a hardware clock. One way to improve the synchronisation between serial data and hardware timed outputs is to use extra hardware to trigger the transmission of serial data. If the trigger is timed using the same clock as other outputs and the transmission delay is accounted for, then serial data can be output more synchronously. The scheme for timing serial messages is

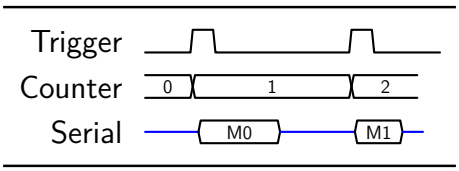

Figure 1.2: Timing diagram for serial communication. A counter channel is configured to count edges from a digital output channel. Every time it sees a rising edge, it triggers the output of the next message on each serial channel from the computer. Multiple messages can be communicated during a single sequence without the need for software timing.

shown in Figure 1.2. Serial messages are stored as strings on the computer and a counter channel is configured so that every time it detects a rising edge, the computer outputs the next message. This counter is connected to a digital output channel, so that it acts as a trigger for the serial data output. Using this method, multiple serial messages can be sent to one device during a sequence even for devices which have no means of storing commands.

1.3.4 Voltage Acquisition

Analogue input channels are configured in a similar way to analogue output channels. A block of memory is allocated on the **daq!** card for each input channel. Once the card is triggered to start acquiring, an **adc!** (**adc!**) converts the voltage across the input into a digital value at every rising edge of the clock signal. Once the sequence has finished, or the buffer has been filled, the card streams this data to the computer.

1.4 MOTMaster Sequences

In addition to interfacing with control hardware, MOTMaster is used to define the structure of experimental sequences. In earlier versions of MOTMaster, sequences were defined using functions within a C# source file. To run an experiment, MOTMaster

compiled this file to build the voltage patterns and wrote them to the hardware. Whilst this had little overhead in resources needed to build and run a sequence, modifying and debugging sequences was much more time consuming. Taking inspiration from Cicero, the user interace of MOTMaster was redesigned so that sequences could be expressed graphically. They are then built using the same functions as before, so that from the point of view of the hardware, the two methods of control are equivalent.

1.4.1 Sequence Structure

A MOTMaster sequence is composed of a list of sequence steps, which define the state of the control hardware over a discrete amount of time. Each step is defined to last for a duration which must be an integer multiple of the timebase (e.g. 10 µs for a 100 kHz sample clock frequency). During a step, a digital channel is either high or low and each serial channel can send one message, which is encoded as a string of text. Analogue output channels can be configured to output a single voltage, step from one value to another, or linearly ramp to a specified value. A sequence step is useful to represent a single action, so that each stage of the experiment, for example the initial **mot!** loading phase, is composed of multiple steps. Numerical values, such as analogue voltages or times, can be represented by named parameters. The value of a parameter can be updated between each cycle of the experiment, so that MOTMaster can implement a scan by iterating a parameter through a range of values. Besides determining the state of the output hardware, the sequence steps are used to define when to acquire from the analogue inputs. A specific digital channel, named `acquisitionTrigger`, is reserved as a start trigger for the acquisition. Analogue data starts being acquired during the step this channel goes high and stops when it goes low. Therefore, the number of samples required depends on the time between the first and last step and the analogue input sample clock frequency.

1.4.2 Running a Sequence

MOTMaster is designed to run in two modes, referred to as repeat and scan. The distinction between these is that the repeat mode does not need to recreate a sequence between each cycle. Before MOTMaster starts controlling the experiment, the sequence is built and the output hardware is configured to regenerate their patterns. In practice, this reduces the delay between each cycle, which is largely a result of the time needed to process acquired data and reconfigure the control hardware. In contrast, scan mode varies a parameter during each cycle, so additional time is required to rebuild the sequence and write to each **daq!** card. Aside from this, these modes operate equivalently. At the start of an experiment cycle, the hardware is initialised and timing properties, such as the trigger and sample clock for each **daq!** card is set. The sequence represented in the user interface is converted into the analogue and digital voltage patterns for each **daq!** card. As mentioned previously, the required buffer for the analogue input data is calculated based on the state of the `acquisitionTrigger` channel. If any serial commands are used, the timing properties of the counter channel are configured, similarly to the rest of the **DAQ!** (**DAQ!**) hardware. The sequence is started by sending a software trigger to one output card, which is configured to export its start trigger to the other cards. This ensures that start of the output of each card is synchronised. After the sequence has finished, any acquired data from the analogue input channels is streamed to the computer. The data per channel are segmented into arrays that were acquired during each sequence step, before additional post-processing if required. Finally, the hardware is reset to its initial state, before starting the next experiment cycle.

1.5 Experiment Control Hardware

In the preceding sections, the discussion of MOTMaster has been presented without referring to specific hardware used in this experiment. Subsequent chapters will introduce components of the experiment that are controlled by a computer, but it is worth introducing the hardware used to implement this control. All of the **daq!** cards are housed on a PXIe 1073 chassis, so that timing signals such as start triggers and sample clocks can be shared on the PXI backplane in order to minimise the number of external cables required to synchronise the devices. The analogue output signals are generated on a PXI-6723 card. This contains 32 analogue output channels and the output of each is generated using a 13 bit **dac!**. Over the maximum voltage range of $\pm 10\,\mathrm{V}$, this corresponds to an output quantisation of 2.44 mV, which did not limit the precision of any analogue control in the experiment. The analogue output pattern is sampled at a frequency of 100 kHz, which gives a minimum resolution of 10 µs. Any jitter on this sample clock did not produce any noticeable effects during the experiment. This card also contains the counter channel used to trigger serial messages.

Two cards on the chassis are able to acquire data from analogue inputs. The first is a PXIe-6341, which has 16 input channels, each with a 16-bit **adc!**. During the preliminary stages of the experiment, this bit-depth was sufficiently large to prevent quantisation effects becoming a dominant source of error. However, as discussed further in Section **??**, the AI Q 2010 MEMS accelerometer used in the experiment has a noise spectral density which necessitates a greater bit-depth **adc!**. Therefore, a PXI-4462 card, which contains 4 24-bit analogue input channels, was added. This card is used to acquire data from devices where the higher voltage resolution is desirable — namely, the MEMS accelerometer and detection photodiode.

Digital output signals are generated using a PXI-6541 card. Unlike the others, this card is controlled using the NI-HSDIO driver. With a maximum sampling frequency of 50 MHz, this card is capable of generating digital signals at a much higher rate than the PXIe-6341, which also contains digital output channels. However, the PXIe-6341 card only contains 8 digital channels that can be timed using a hardware clock, fewer than required to control the entire experiment. A second, historical, reason to use a card with greater timing resolution was to reduce the effects of timing inaccuracies on the interferometer phase. As discussed in Section **??**, a difference in the time between each interferometer pulse results in a phase difference which is independent of inertial forces and reduces sensitivity. Before the M Squared laser system was developed, the original plan was to use the μQuans laser to provide light for the interferometer and control the laser pulses using **ttl!** (**ttl!**) switches to an **aom!** and a shutter. In this case, the greater timing accuracy of the PXI-6541 was desirable. Since the M Squared laser contains a dedicated module to synthesise analogue and digital waveforms to control the interferometer pulses, this requirement on the timing accuracy of the digital channels controlled using MOTMaster is no longer necessary.

Two components of the experiment are controlled during the experiment using serial communication. The first of these is an interface to the **dds!** (**dds!**) on the μQuans laser which control the frequency of the cooling and repump lasers and is controlled in real-time during the experiment. This communication protocol is described in further detail in Section **??**. In addition to this, a serial interface is used to control the frequency of a WindFreak microwave synthesiser. Its use in the experiment is described in Section **??**. Unlike the μQuans laser, no serial data is sent to the WindFreak after the sequence starts. The output frequency is set between cycles and remains constant throughout. Finally, MOTMaster is configured to remotely connect to the M Squared laser, so that it can control all the parameters necessary to drive Raman transitions during the experiment. This is done by sending structured JSON messages

that contain commands to implement this control. More detail on how this is used in the experiment is given in Section **??**.

Bibliography

[1] Aviv Keshet and Wolfgang Ketterle. "A Distributed GUI-based Computer Control System for Atomic Physics Experiments". In: (2012). DOI: 10.1063/1.4773536. arXiv: 1208.2607.