User Manual

Zhaohui Yang, [zhy@arizona.edu](mailto:zhy@arizona.edu)

[Quantum Information and Materials Group](https://quantum.lab.arizona.edu/), University of Arizona

*Last updated: May 2022*

This manual consists of the design framework, implementation details, usage examples, necessary configuration descriptions and API documentation of the GUI software for ODMR experiments.

Contents

[User Manual 1](#_Toc104339743)

[Design Framework 3](#_Toc104339744)

[Hardware and software support 3](#_Toc104339745)

[Odmactor SDK design 4](#_Toc104339746)

[OOP implementation 4](#_Toc104339747)

[Scheduler classes 6](#_Toc104339748)

[Modules structure 7](#_Toc104339749)

[GUI overview 8](#_Toc104339750)

[Operation Guiding 9](#_Toc104339751)

[Odmactor-GUI functionalities 9](#_Toc104339752)

[Configuration list 9](#_Toc104339753)

[Demonstration pages 13](#_Toc104339754)

[Frequency-domain ODMR 15](#_Toc104339755)

[Continuous-wave ODMR detecting 15](#_Toc104339756)

[Pulse ODMR detecting 16](#_Toc104339757)

[Time-domain ODMR 18](#_Toc104339758)

[Rabi oscillation 18](#_Toc104339759)

[Ramsey detecting 19](#_Toc104339760)

[T1 relaxation 20](#_Toc104339761)

[Hahn echo 21](#_Toc104339762)

[High-order dynamical decoupling 21](#_Toc104339763)

[Other scheduling modes 22](#_Toc104339764)

[Specific Considerations and Future Functionalities 22](#_Toc104339765)

[Specific considerations 22](#_Toc104339766)

[MW on/off control 22](#_Toc104339767)

[Customized scheduler 22](#_Toc104339768)

[Frequency shift phenomenon 22](#_Toc104339769)

[Future functionalities 23](#_Toc104339770)

# Design Framework

The initial goal of this project is to provide an as universal as possible SDK of the series of usual scheduling algorithms for [ODMR](https://en.wikipedia.org/wiki/Optically_detected_magnetic_resonance) and spin manipulation experiments. Currently the SDK project (named [Odmactor](https://github.com/Youngcius/odmactor)) supports all the frequency-domain and time-domain ODMR operation functionalities and all of them have been applied in the [QIM](https://quantum.lab.arizona.edu/) group at UA. Furthermore, a desktop software with user-friendly GUI building on this SDK and Qt graphic library was also implemented and tested well in QIM lab. It is called [Odmactor-GUI](https://github.com/Youngcius/odmactor-gui) in GitHub. Following content includes descriptions of the algorithms design and GUI implementation as well as the integration approach.

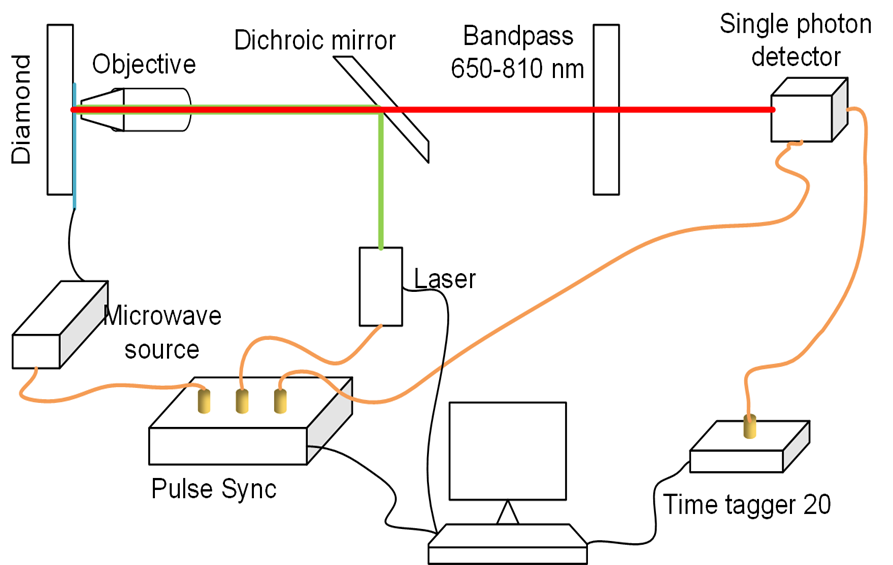
## Hardware and software support

This project is designed for physical experiments, thus it does not depend on only software techniques. The following figure shows usual necessary devices of a ODMR platform:

1. Microwave source (MW): also called RF signal generator. The [SMB 100A](https://www.rohde-schwarz.com/us/products/test-and-measurement/analog-signal-generators/rs-smb100a-microwave-signal-generator_63493-9379.html) provide by [R&S](https://www.rohde-schwarz.com/) is what we used currently.
2. Arbitrary sequence generator (ASG): the most crucial device, used for generating control sequences. Here in we use the ASG8005 provided by [CIQTK](https://en.ciqtek.com/)
3. Avalanche photodiode (APD) + Time Tagger: the most usual photon counting scheme. We use a semiconductor APD and an [8-channel Time Tagger](https://www.swabianinstruments.com/time-tagger/) provided by [Swabian](https://www.swabianinstruments.com/).
4. Lock-in amplifier + DAQ: another data acquisition scheme. When signals are weak, the APD + Tagger cannot detect effective signals, we use this lock-in scheme for detecting. The lock-in amplifier we use is [RS830](https://www.thinksrs.com/products/sr810830.htm) provided by [Stanford Research](https://www.thinksrs.com). The DAQ is a regular device provided by [National Instruments](https://www.ni.com).
5. Laser: 532nm laser or 730nm laser. In the detecting on [NV in diamond](https://en.wikipedia.org/wiki/Nitrogen-vacancy_center), we use the 532nm fiber pulse laser provided by CIQTK; in the detecting on [SiV in SiC](https://www.nature.com/articles/s41467-019-09873-9), we use the 730nm free-space solid laser. They are both modulated by ASG’s output sequences via connection to the laser itself or acoustic optical modulator (AOM).

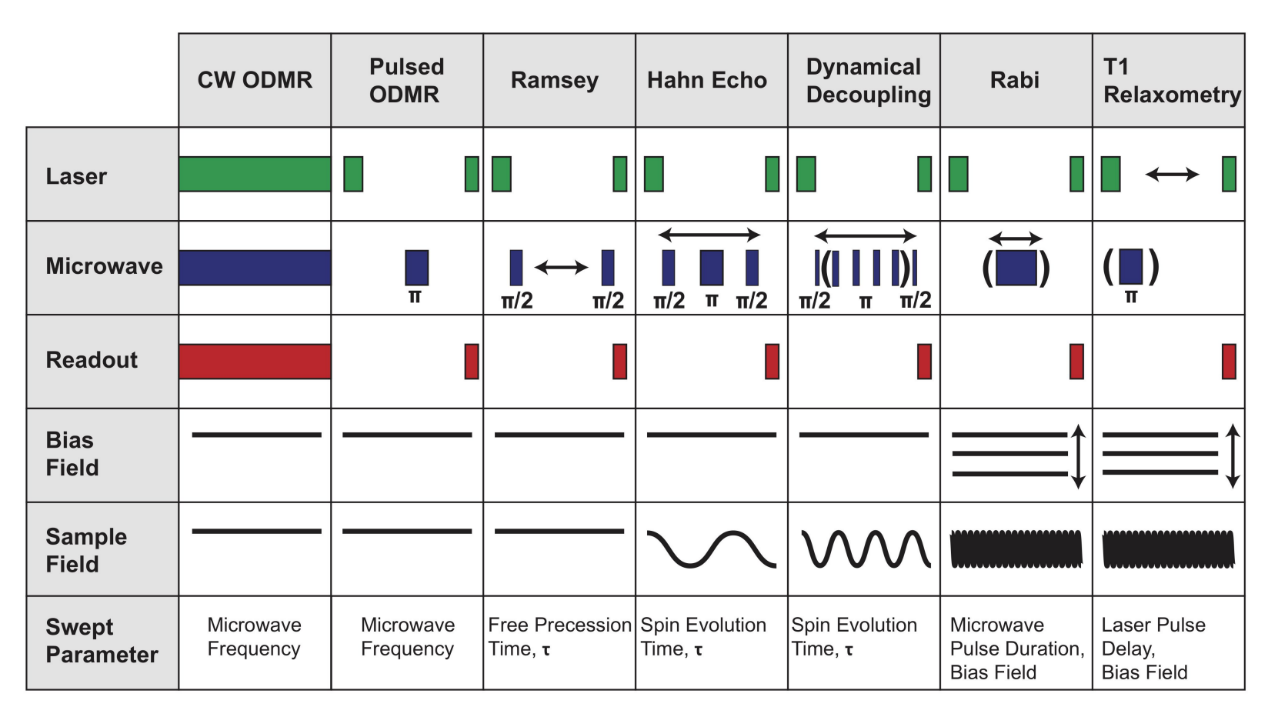
Except for the [Time Tagger](https://www.swabianinstruments.com/static/documentation/TimeTagger/index.html), all the corresponding instrument vendors do not provide user-friendly programming interfaces. To use them in an autonomous manner, we have to program from scratch, building on libraries related to soft-hardware interfaces, e.g., pymeasure, pyvisa, RsInstrument. The core soft-hardware programming is on ASG, via a compiled module provided by CIQTK.

Other necessary libraries are those standard and usual third-party ones. All the dependencies are listed in file requirements.txt.



## Odmactor SDK design

### OOP implementation



This SDK is an OOP Python project. It mainly implements a set of schedulers by abstracting the universal pipelines and usual settings of relevant ODMR experiments. Usually, ODMR experiments are divided into “frequency-domain” and “time-domain” two detecting modes, as shown in the figure below. Both needs Laser/MW/ASG sequences settings and photon counts readout. In the former mode, the MW frequency needs to change to acquire a ODMR spectrum. In the latter mode, what needs to be changed is time, which can be MW operation time or spin evolution time or other delay times. Thus, in an OOP insight, they are implemented in a set of “scheduling” classes with well-defined inheritance relationships.

Technically, the time-domain measurements are not called ODMR experiments, while they are performed in the ODMR platform. Therefore, we unitedly call them ODMR measurements. Usages of them are briefly summarized as follows.

*Frequency-domain ODMR:*

1. Continuous-wave detecting: measure rough ODMR spectrum
   1. Polarize spin systems
   2. Operate continuous MW and readout signals for the whole sequence period
2. Pulse detecting: measure fine ODMR spectrum
   1. Polarize spin systems
   2. Apply a fixed-interval MW pulse
   3. Readout final fluorescence signals

*Time-domain ODMR:*

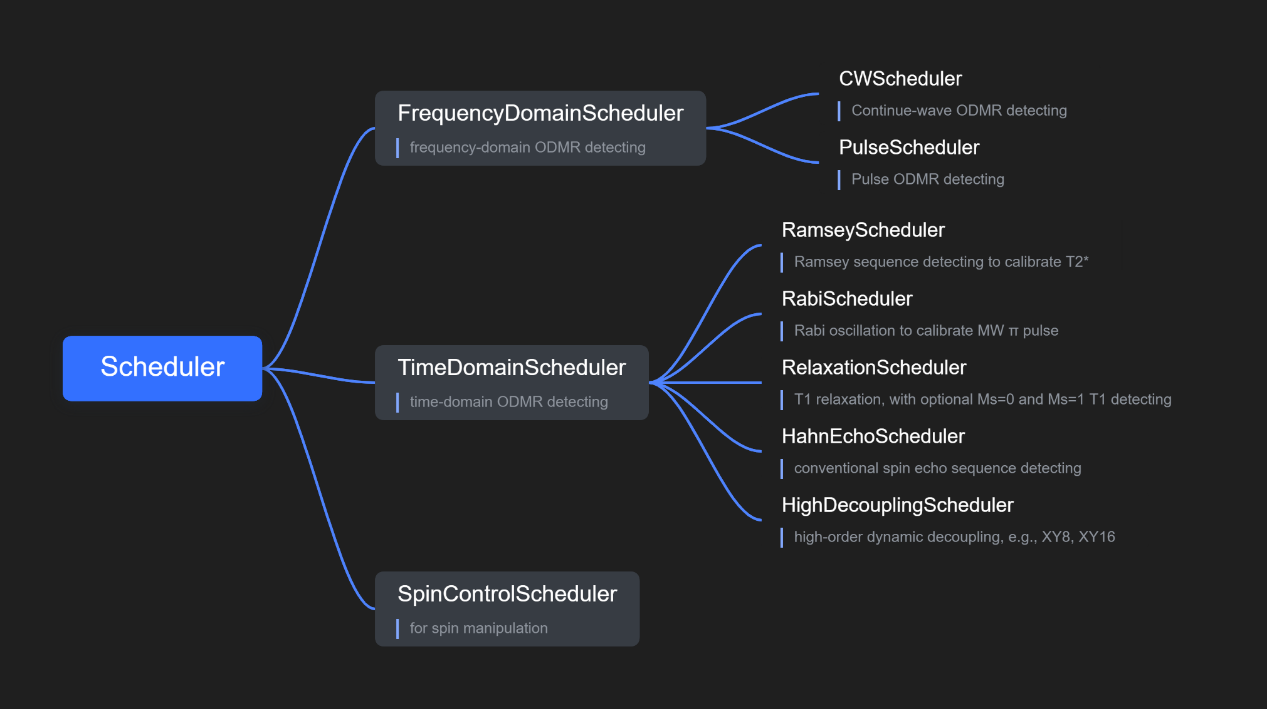
1. Rabi oscillation: calibrate MW pulse
   1. Initialize spin systems to ground states
   2. Operate a MW pulse with variable interval
   3. Readout final spin states (population), i.e., fluorescence signals in ODMR
2. Ramey detecting: measure T2\* of spin systems (dephasing time)
   1. Initialize spin systems to ground states
   2. Initialize spin systems to equal-amplitude superposition state using the calibrated MW pulse
   3. Wait for a time interval
   4. Operate a calibrated MW pulse again
   5. Readout final spin states
3. T1 relaxation: measure spin-lattice relaxation time
   1. Initialize spin states to ground states
   2. Initialize spin states to excited states using the calibrated MW pulse (optional)
   3. Wait a time interval
   4. Readout final spin states
4. Hahn echo: measure T2 of spin systems (decoherence time)

Sequences are similar with those for Ramsey detecting, while there is an additional MW pulse between the two pulses.

1. High-order dynamical decoupling: extend the decoherence time

Sequences are similar with those for Ramsey detecting, while there are additional MW pulse between the two pulses.

### Scheduler classes



As the OOP modeling insight, the series of scheduler classes are implemented with such inheritance relationships. Classes Scheduler, FrequencyDomainScheduler, TimeDomainScheduler, and SpinControlScheduler are all abstract base classes, which cannot be directly used by users. Instead, user should use the application-level classes (CWScheduler, PulseScheduler, HahnEchoScheduler, etc.) to run ODMR scheduling pipelines. Notably, the abstract class SpinControlScheduler is reserved for fine spin manipulation. It might be required for future quantum information processing experiments building on single spin systems. Due to current restricted experimental conditions and requirements, it is not yet implemented.

A picture containing table

Description automatically generated

The following structure demonstrates part of the necessary data fields, general scheduling methods (methods of the base class) and specific scheduling methods. By means of the SDK, there is a set of standard scheduling pipelines, for which please refer the [example files](https://github.com/Youngcius/odmactor/tree/master/example) or the [README of Odmactor](https://github.com/Youngcius/odmactor/blob/master/README.md).

### Modules structure

This Python SDK is composed of three packages scheduler, instrument, utils. Package scheduler consists of corresponding modules which are the implementation of the series of scheduler classes above. Package scheduler is the software-level abstraction of instruments, including ASG, MW, Lock-in Amplifier, Time Tagger, etc. Although for some of them, instrument providers have provided user-friendly SDK or programming guidance, the usage is not convenient and more importantly, not consistent. Herein we construct abstract instrument classes with universal operation methods, i.e., the instruments can be operated in an OOP approach. For instance, all the abstract instrument classes in our project consist of the same-name member methods like start, stop and close. And each instrument is automatically connected once its corresponding software instance is initialized. Package utils consists of utils functions necessary in this SDK, particularly the functions for processing ASG sequences. In the GUI software, another package ui consists of predefined UI description files (“.ui” and “.py”), implemented drawing on [QtDesigner](https://doc.qt.io/qt-5/qtdesigner-manual.html) and [PyUIC](https://pypi.org/project/pyuic5-tool/).

Each module and each function in it consists of detailed docstrings and even parameter type declarations. User can reach to the source code or generate vivid Sphinx-style API documentation.

├─instrument  
│ │ asg.py  
│ │ laser.py  
│ │ lockin.py  
│ │ microwave.py  
│  
├─scheduler  
│ │ base.py  
│ │ customization.py  
│ │ decouple.py  
│ │ frequency.py  
│ │ spin.py  
│ └─ time.py  
│  
└─utils  
 │ plotting.py  
 │ sequence.py  
 │ utils.py  
 │  
 ├─asg  
 │ ASG8005\_PythonSDK.py  
 │ ASGDLL\_x64.dll  
 └─ ASGDLL\_x86.dll

## GUI overview

A screenshot of a computer

Description automatically generated

The main body of this UI consists of “configuration list” and “demonstration pages” two parts, which are implemented based on “Tool Box” and “Tab Widget” Qt components, respectively. In the “configuration list” part (blue framed), user can perform generic operations on instruments, configure ODMR detecting sequences and scanning range, and perform spin manipulation (not implemented yet). In the “demonstration pages” (green framed), use can have a look at real-time photon counting, ODMR detecting results, and sequences visualization. Details of them are described in the next section.

# Operation Guiding

## Odmactor-GUI functionalities

Once familiar with the entire ODMR operation mode, functionalities of corresponding components (widgets) are demonstrative. There are still some components worthy additional descriptions.

### Configuration list

*MW generic settings:*

Graphical user interface, application

Description automatically generated

pulse is not necessary to be set except for Ramsey/HahnEcho/HighDecoupling modes.

When “timer” is checked, MW will automatically stop after a set time interval once the push button “on/off” is checked.

*Laser generic settings:*

Graphical user interface

Description automatically generated

Current Lasers do not support direct programming control. But if they do, required control functionalities should be as the above components show. It is a reserved UI module.

*ASG channels settings:*

Graphical user interface

Description automatically generated

Our ASG has 8 output channels. Laser, MW, APD and Tagger are controlled by ASG in most cases. ASG can also generate lock-in synchronic sequences to MW and Lock-in Amplifier, but usually it is not economic. Instead, user often use an outside signal generator to generator lock-in synchronic signals. All control sequences can be set to high-effective or low-effective arbitrarily.

*Time Tagger input channels:*

Table

Description automatically generated with low confidence

Out Tagger has 8 input channels that can be fed by detected signals or trigger signals. The input channel settings are nontrivial only if the radio button “Use Tagger” is checked. Otherwise, the backend detecting is finished by Lock-in Amplifier + DAQ.

*ODMR operations:*

Graphical user interface, text, application

Description automatically generated

These three buttons, when user wants to start a ODMR pipeline, after configuring parameters, one should always click “Load Sequences” and “Start Detecting” by order. “Load Sequences” will trigger the built-in scheduler read parameters from UI components and load sequences to ASG hardware as well as visualize them. “Start Detecting” will let the built-in scheduler start running in a scanning frequency/time mode and visualize results in real time. Another button “Save Data” is used to manually save ODMR measurement results into the default “Downloads” folder of Windows system. In fact, without manually clicking the “Save Data” button, results are still saved into disk automatically.

*Frequency-domain scanning settings:*

Graphical user interface, text, application, email

Description automatically generated

Two selectable detecting modes: CW, Pulse

“Scanning Frequency” checkable box: checked in default. If not checked, the scheduling will run on a single frequency point and save the results automatically, but without data visualization. Currently the single step running functionality is not implemented, so please do not de-check this checkable box.

Scanned MW frequencies configuration: by setting “start”, “end” and “step”, an isometric scanned frequencies list will be generated automatically

*Time-domain scanning settings:*

Graphical user interface, text, application

Description automatically generated

Five selectable detecting modes: Rabi, Ramsey, T1 relaxation, Hahn echo, High-order dynamical decoupling. Only for the “High-order DD” mode, the spin box (“Order”) is nontrivial.

“Scanning Time” checkable box: similar with that for frequency-domain settings.

Scanned delay time intervals configuration: by setting “start”, “end” and “step” (or “length”), a scanned frequencies list will be generated automatically. For “Step mode”, only “start”, “end” and “step” inputted are effective; for “Length mode”, only “start”, “end” and “length” inputted are effective; for “Logarithm length”, only “start”, “end” and “step” inputted are effective, but the generated intervals list is not isometric but exponential isometric, which is useful for long-time experiments like T1 measurement.

*ODMR sequences configuration:*

Graphical user interface, table

Description automatically generated

The following sketch demonstrates meanings of most of the above sequence parameters (“inter” means “interval”; “init” means “initialization”).

**asg laser channel:**  
-------------- --------------------------------------------------------------  
|(laser init)| | |  
| |------------------------------------------------| |

**asg microwave channel:**  
 ----------------  
 (init-mw inter)|(mw operation)|(mw-read inter)   
-----------------------------| |--------------------------

**asg tagger acquisition channel:**  
 ------------ ------------  
 (pre-read)|(sig read)|(read inter)|(ref read)|(period inter)  
-------------------------------------------------------------------------| |------------| |--------------

Graphical user interface

Description automatically generated

For each detecting setting point, i.e., one specific frequency or time interval setting, the ODMR experiment requires to repeat sequences for periods. For each period, there is one pulse readout operation in Tagger, whose readout window is defined by the Laser readout pulse interval. Repetition is for stable and reliable results. For instance, sequence in one period like the above figure (without lock-in synchronic sequences visualization) will be repeated and thus sequences like the following figure (channel 4 and 8 means lock-in synchronic sequences) are acquired (x-axis ticks are trivial). In actual experiment, single period is much short relative to the lock-in synchronic period.

Chart

Description automatically generated with medium confidence

In the CW mode, “period” is determined by either the “lock-in sync frequency” (Lock-in mode) or “Laser initialization” (Tagger mode) parameter. Otherwise, pulse ODMR sequences should be configured by setting detailed parameters, in which “Laser initialization”, “Microwave operation” and “Signal readout” are always necessary.

### Demonstration pages

*Photon counting:*

Whether data is acquired by APD + Tagger or Lock-in Amplifier + DAQ, this page always supports real-time photon counts readout and visualization. Components settings in the figure are obvious. “Connect tagger” and “Close tagger” buttons are nontrivial only in Tagger counting mode.

Chart, bar chart

Description automatically generated

*ASG sequences settings and visualization:*

Functionalities of this page are similar with the native software provided by CIQTK but with a different sequences configuration logic. Better yet, our setting manner is more convenient and more robust. That is supported by the extra sequence module in Odamctor SDK.

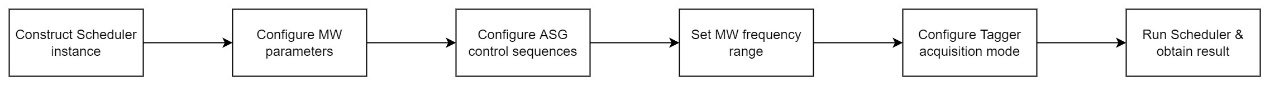
Our ASG device has 8 output channels, which corresponds to the 8-row table in the following figure. Each sequences data are set according to “High-Low-…-High-Low” list rule. For example, [100, 0] means a continuous high-level sequence, while [0, 100] means a continuous low-level sequence. All the data unit is “ns” in default.

The following figure shows generated sequences are aligned automatically as well as the corresponding chart.

Chart

Description automatically generated

## Frequency-domain ODMR



All the frequency-domain ODMR experiments follow such a pipeline, whether user is using the SDK or the GUI software.

The following examples are about testing the SiV in SiC, whose zero-field splitting (ZFS) of the ground spin state is 70MHz. And due to the weak photon signals of SiV, Lock-in Amplifier is used.

### Continuous-wave ODMR detecting

1. Set MW: power dBm
2. Select “CW” detecting mode
3. Set scanned frequencies: , , (MHz)
4. Set sequences repetition period: ( lock-in sync periods only for CW lock-in mode; in default)
5. Set lock-in sync frequency: 50Hz (in default)
6. Click “Load Sequences” button (sequences chart is as follows)

Chart, bar chart

Description automatically generated

1. Click “Start Detecting” button
2. Wait and observe real-time result in the “Frequency-domain detecting” tab page (spectrum is as follows)

Chart, line chart

Description automatically generated

1. Detailed results are saved to “Downloads” folder automatically in “.json” format

### Pulse ODMR detecting

1. Set MW: power dBm
2. Select “Pulse” detecting mode
3. Set scanned frequencies: , , (MHz)
4. Set sequences repetition period: ( is repetition number each pulse sequences period)
5. Click “With Reference Signal” and “With MW On/Off” two checkable boxes (the following figure is a spectrum with green-curve reference signals)
6. Set lock-in sync frequency: 50Hz (in default)
7. Set pulse intervals: , , , , (ns)
8. Click “Load Sequences” button (sequences chart is as follows)

Chart

Description automatically generated

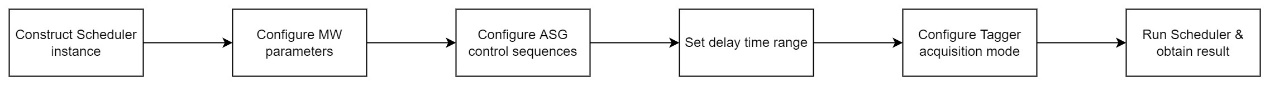
1. Wait and observe real-time result in the “Frequency-domain detecting” tab page (spectrum is as follows)

Chart, line chart

Description automatically generated

1. Detailed results are saved to “Downloads” folder automatically in “.json” format

## Time-domain ODMR



Similarly, all the time-domain ODMR experiments follow such a pipeline.

### Rabi oscillation

First place a permanent magnet to create a static magnetic field around spin defects. Then use “CW” or “Pulse” ODMR measure one of the spectral peaks, which is about 47MHz. Then conduct usual Rabi oscillation experiments.

Chart, line chart

Description automatically generated

1. Set MW: power dBm, frequency MHz
2. Select “Rabi” detecting mode
3. Set scanned time intervals: , , (ns)
4. Set sequences repetition period: N=10000 (N is repetition number each pulse sequences period)
5. Set lock-in sync frequency: 50Hz (in default)
6. Set pulse intervals: , , , (ns)
7. Click “Load Sequences” button (sequences chart as follows is just a sketch; in fact, the MW operation interval is variable)

Chart

Description automatically generated

1. Wait and observe real-time result in the “Time-domain detecting” tab page (spectrum is as follows)

Chart, line chart

Description automatically generated

1. Detailed results are saved to “Downloads” folder automatically in “.json” format

### Ramsey detecting

Ramsey detecting depends on the calibrated MW pulse parameters. From the above figure, we roughly regard the MW pulse configuration is: MHz, ns, dBm.

1. Set MW pulse parameters as above
2. Set MW: its frequency and power can be the same as the above pulse; otherwise, the actual applied pulse parameters will be regulated automatically
3. Select “Ramey” detecting mode
4. Set sequences repetition period: N=10000 (N is repetition number each pulse sequences period)
5. Set lock-in sync frequency: 50Hz (in default)
6. Set pulse intervals: , , , (ns)
7. Click “Load Sequences” button (sequences chart as follows is just a sketch; in fact, the interval between two pulse is variable)

Chart

Description automatically generated

1. Wait and observe real-time result in the “Time-domain detecting” tab page (spectrum is as follows)
2. Detailed results are saved to “Downloads” folder automatically in “.json” format

### T1 relaxation

For classical ground-excited spin systems, there are two T1 relaxation measurement modes:

1. The initial is ground state: The entire period is without MW pulses
2. The initial is excited state: Each period is with a MW pulse after Laser initialization

Herein we demonstrate the initial-ground-state T1 relaxation. The operation pipeline is like Ramsey detecting. Sequences generated are as follows.

Chart

Description automatically generated

### Hahn echo

Check the radio button “Hahn echo”. Other steps are the same as Ramsey detecting.

Chart

Description automatically generated

### High-order dynamical decoupling

Check the radio button “High-order DD”. Set desirable “decoupling order”. Other steps are the same as Ramsey detecting.

User also need to set the “decoupling order”. When the order is 1 (the default value), it is equivalent to Hahn echo measurement; when the order is 0, it is equivalent to Ramsey measurement. The following figure shows an instance of 8-order decoupling sequences.

Chart

Description automatically generated

## Other scheduling modes

This software also supports customized scheduling modes. User just needs to set sequences manually in the “ASG Sequence” tab page. Other settings like MW and are similar with other ODMR scheduling pipelines.

# Specific Considerations and Future Functionalities

Th above functionalities can satisfy all current requirements of QIM group. But some configuration rules are made due to our hardware restrictions. And there are some TODO items in the near or long term.

## Specific considerations

### MW on/off control

When detecting the reference signals, MW needs to be open/closed alternately. For fine manipulation, it is realized by open/close RF Switch by ASG. While the testing results on May 23 shows there are issues of the Switch. So user needs to realized reference signals detecting by directly open/close MW itself, i.e., the checkable box “With MW On/Off” needs to be checked when “With Reference Signal” is checked.

### Pulse readout

When using Lock-in Amplifier + DAQ, ASG cannot control the amplifier to readout in a very short pule window (~102 ns). Actually the ultimate results is a stable-level signals. That is, the scheduler will run for periods and then perform one-time signals readout. On the other hand, reference signals detecting with lock-in scheme is not theoretically baseless.

When using APD + Time Tagger, the pulse readout is realizable, controlled by ASG trigger sequences (channel 5 in our example). For reference signals readout, the best manner is two-pulse readout in each period. But there is results unreliability with our Time Tagger device. Therefore, we alternatively realize reference measurement by reading -period signals continuously and then reading -period reference signals continuously. The final results are aggregated by averaging.

### Frequency shift phenomenon

When using Lock-in Amplifier + DAQ, there sometimes exists frequency shift on the measured ODMR spectrum (the left figure below). That is because there is a transition time for the lock-in amplifier's counts to be stable. When the integration time of each scanned frequency point is enough, this phenomenon does not exist (the right figure below). More exactly, the relative time, i.e., integration time(Odmactor) v.s. time constant(Lock-in Amplifier device panel) shall be long enough.

Chart, line chart

Description automatically generated Chart

Description automatically generated

## Future functionalities

### Spin manipulation

There is a reserved “Tool Box” column named “Spin Manipulation”, which will be most essential module for future requirements. QIM group aims to ultimately realize single spin control on color defects. Single spin manipulation is the core of solid-state quantum information processing experiments. In that case, usually only MW pules are necessary. Because QIM lab has not the corresponding hardware platform and practical requirements, related paradigms have not been abstracted into programming implementation.

### Customized scheduler

Testing and debugging for customized schedulers have not been done. And some implementation details need to be modified. So user shall not use this kind of mode.

### Single step running

As mentioned above, the single step running functionality is not implemented now, so please do not de-check the “Scanning Frequency” and “Scanning Time” checkable boxes.