Deposition Software for the Inficon IC6 Deposition Controller[*]

Marco Esters
University of Oregon
esters@uoregon.edu

June 14, 2017

Contents

[*]This documentation refers to version 1.0.2, last revised on 05/29/2017. Published under the GPL 3.0 license.

1. Introduction

The IC6 is a fully programmable deposition controller developed by Inficon. It is a powerful device and has a lot of functionality, which provides many opportunities, but also requires extensive programming effort to set up simple depositions.

This program provides an easy to use interface to deposit arbitrary layering sequences with an IC6 deposition controller. It was primarily designed to synthesize precursors for the Modulated Elemental Reactants (MER) method[1], specifically for ferecrystals[2], but it can be used for the preparation of other materials.

This documentation begins with the installation and configuration procedure for the deposition program, followed by a short user's guide on how to set up and start a deposition in section 3. These two sections are all that should be necessary to operate the software.

If there are problems with the IC6, the user may need to interact directly with the IC6. For that purpose, section 4 describes how to use the software to send commands directly to the IC6 and how to interpret its responses, especially its error codes.

Finally, section 5 gives some information for developers who may want to modify or expand the code. This section is intended to supplement the extensive comments in the block diagrams, and describe the purpose of each VI, variable and file (section 5.1), a more detailed explanation on the communication protocol (section 5.2), and the implemented deposition algorithm (section 5.3).

My thanks go out to "Team 307" for extensively testing the software on their system. I also appreciate their patience for letting me hijack their system while I was just doing "cosmetic work". I also thank Danielle Hamann, Alex Lygo, and Liese Maynard from the University of Oregon for proofreading this documentation.

The software has been extensively tested on the deposition system in the Dave Johnson group in the Department of Chemistry at the University of Oregon using IC6 version 1.20. Despite my efforts to eliminate all bugs, there is of course no guarantee that the software is free of errors. Please send bug reports to esters@uoregon.edu. Feel free to contact me if you find mistakes or unclear passages in the documentation.

While the software should work regardless of employed deposition hardware, there is no guarantee it will work on every system. If there are any problems, feel free to contact me. If it works perfectly, I would be happy to hear about it as well – I only got to test it on one deposition system, so I am curious about how it works on others.

[1]DOI: 10.1016/S1359-0286(98)80082-X
[2]DOI: 10.1088/0268-1242/29/6/064012

2. Setup

2.1. System Requirements

The newest version of the software can be downloaded from github[3], and runs without installation.

The software was tested and worked on a Pentium 4 with only 1 GB of RAM, so the minimum hardware requirements are unknown. The only essential hardware is an RS-232 port that is connected to the IC6 with a serial connection. TCP/IP connections are not supported yet. It is recommended to use a screen with a resolution of at least 1600×900 pixels.

The software requires LabVIEW 2012 SP1 or compatible to run. In order to interface with the IC6, the Measurement and Automation Explorer(MAX)/NI-VISA and the National Instruments device drivers, specifically the VISA drivers, are required.

2.2. Software Configuration

A few additional steps are required to configure the IC6 and the software. First, it needs to be checked that MAX/NI-VISA can detect the IC6 and that it is properly configured. Typically, only the baud rate needs to be adjusted – the IC6 manual describes how to check and change the baud rate on the IC6 (the default on the IC6 is 115200). The exact baud rate is not important, the rate set on the IC6 and in MAX/NI-VISA just need to match.

The next step is to make sure that this program can communicate with the IC6. Open the file `write_read_ic6.vi` and change the COM1 port circled red in Figure 2.1 to the port that the IC6 is connected to. Save and close the file, and open `command_send.vi`.[4] Write “H 1” (without quotes) into the text field on the right, and run the VI. The output should be:

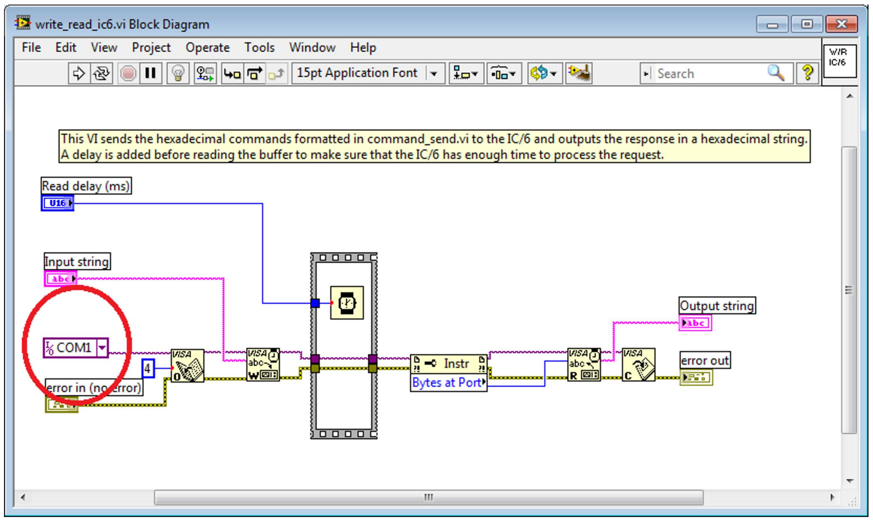
```
1400 00XX 0649 4336 2056 6572 7369 6F6E 20YY 2EYY YY00 XX
```

where the `XX` characters have different values every time the VI is run. The `YY` characters depend on the IC6 version. For version 1.20, it should read `312E3230`. If the output is much shorter, there is an error in the command, or the IC6 is not properly connected or configured. See section 4.3 to interpret IC6 error codes.

The last step in the configuration is to set the active process. For this program to work, one process in the IC6 needs to be reserved. This process must not

[3] `https://github.com/marcoesters/deposition_IC6/`

[4] For a more detailed description, see section 4.1



This VI sends the hexadecimal commands formatted in command_send.vi to the IC/6 and outputs the response in a hexadecimal string.
A delay is added before reading the buffer to make sure that the IC/6 has enough time to process the request.

Figure 2.1: The block diagram of `write_read_ic6.vi`. The element to edit is circled in red.

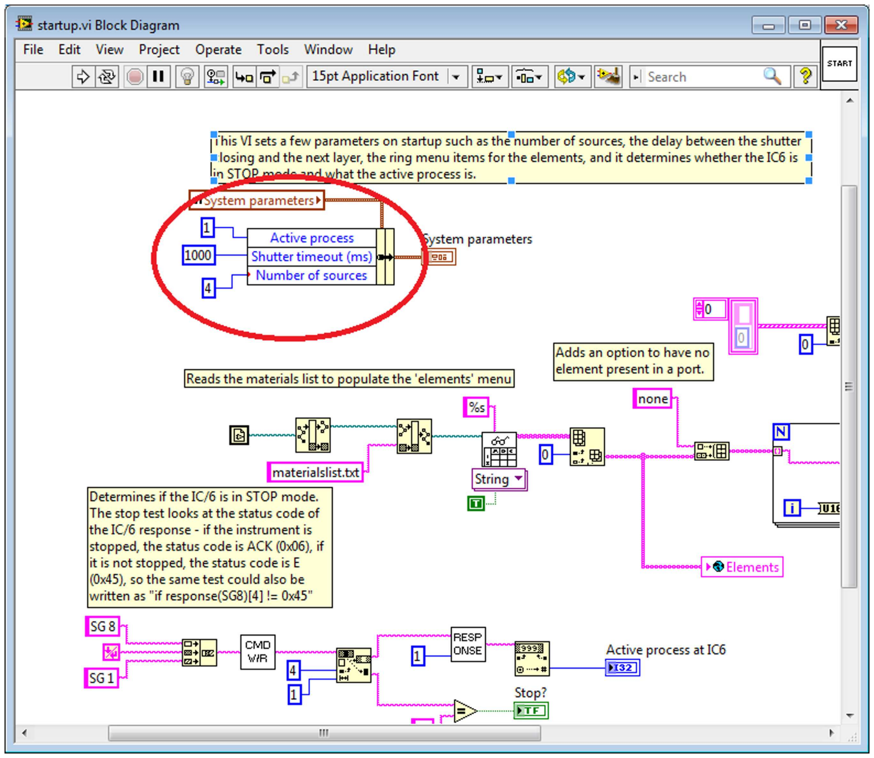
be altered manually, or the software will not work properly. After designating such a process, the value that connects to "Active process" in the block diagram of `startup.vi` (see Figure 2.2) needs to be changed to the designated process number.

The other parameter of interest is "Shutter timeout (ms)". It designates the delay between each deposited layer, i.e. the time the software waits before it opens the source shutters of a new layer after the source shutters of the layer are closed.

The parameter "Number of sources" changes the number of deposition sources recognized by the program. However, changing this parameter is not enough to adjust the number of sources, the user interfaces need to be changed as well. This should be performed by someone with some LabVIEW experience.

In `main.vi`, the arrays with the elements and rates need to be expanded so that the number of rows are the same as the number of sources. In `sequence.vi`, the number of columns of the array for the elements needs to be changed to be equal to the number of deposition sources (or smaller if co-deposition needs to be restricted). These changes will require that some buttons are rearranged and some clusters expanded, which is purely cosmetic and does not interfere with the functionality of the program.

More elaborate changes are required for the VIs `manual_control.vi` and `deposit.vi`. In each VI, the entire cluster, in which the source outputs reside, needs to be expanded first to fit a new set of source outputs. Then, one of the quadrants needs to be selected, copied and pasted into the front panel. This new



This VI sets a few parameters on startup such as the number of sources, the delay between the shutter closing and the next layer, the ring menu items for the elements, and it determines whether the IC6 is in STOP mode and what the active process is.

Reads the materials list to populate the 'elements' menu

Adds an option to have no element present in a port.

Determines if the IC/6 is in STOP mode. The stop test looks at the status code of the IC/6 response - if the instrument is stopped, the status code is ACK (0x06), if it is not stopped, the status code is E (0x45), so the same test could also be written as "if response(SG8)[4] != 0x45"

Figure 2.2: The block diagram of `startup.vi`. The values to edit are circled in red and represent the active process, the time delay between layers, and the number of sources.

set of source outputs then needs to be dragged and dropped into the cluster. It is important that it is inside the cluster with the other source outputs and not on top of it (the frame of the cluster should show a dashed line before dropping the new set out source outputs).

3. Using the Software

3.1. Starting Up the Deposition System

To start the software, open `main.vi` and click on the arrow in the toolbar. If the IC6 is in the STOP state, the VI will automatically start the IC6.

⚠ Pushing the STOP button on the IC6 panel at any time will immediately quit the software.

The VI will also check if the active process on the IC6 is the same as configured in `startup.vi` (see section 2).

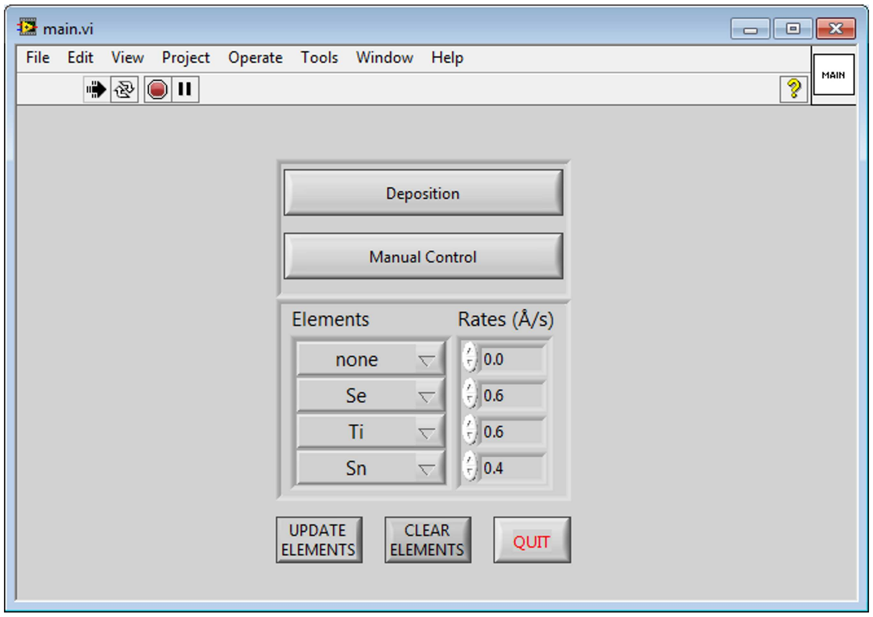
⚠ If the active process is not the same, the deposition software will not work and a popup message will appear. Either quit the software and set the active process in `startup.vi` or on the IC6, or click “OK” and the software will set the active process on the IC6 manually, but it will turn off all deposition sources.

The first step is to set the elements in the deposition sources and the desired deposition rates as outlined in Figure 3.1. If a deposition source is unoccupied or not needed for a depositon, the element can be set to “None”. Pressing the “Update elements” button sends the data to the IC6. The “Clear Elements” button sets all elements to “None” and all rates to 0.0 Å/s on the user interface.

The next step is to heat up the deposition sources until a stable evaporation at the desired deposition rate is achieved. To start this process, push the “Manual Control” button and a new VI will open (Figure 3.2). This will also send the current elements and deposition rates to the IC6 as if the “Update Elements” button was pushed.

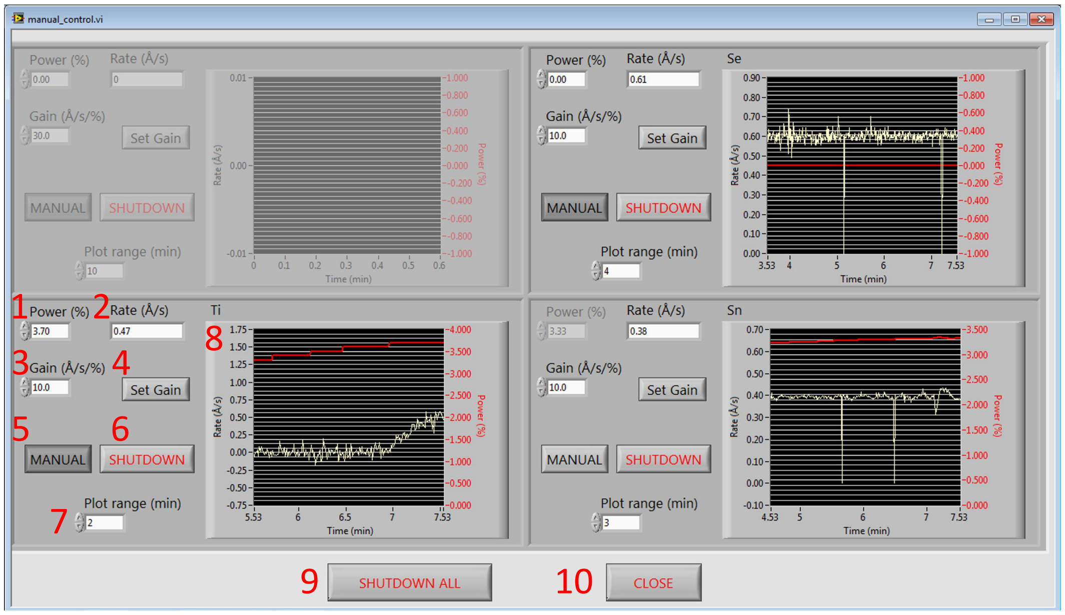
With this VI, the deposition sources can be heated up to the desired deposition rate, provided that the source is connected to one of the DAC outputs at the back panel of the IC6. It also requires the deposition sensor to be connected and assigned to the DAC output at the IC6. Section 4 in the IC6 manual describes in detail how to connect deposition sources and sensors.Sources that are not controlled by the DAC output cannot be heated up with this VI, but the deposition rate can still be monitored if the sensor is attached to the IC6.

The VI has several control and output elements as shown in Figure 3.2:



Elements	Rates (Å/s)
none	0.0
Se	0.6
Ti	0.6
Sn	0.4

Figure 3.1: The front panel of `main.vi`



	Power (%)	Rate (Å/s)	Gain (Å/s/%)	Plot range (min)
	0.00	0	30.0	10
Se	0.00	0.61	10.0	4
Ti	3.70	0.47	10.0	2
Sn	3.33	0.38	10.0	3

Figure 3.2: Front panel of the VI to manually control the deposition sources. For a description of each control and output element see text.

1. **Power (%)** changes the source output power if in the Manual state. If in Non-Deposit Control state, it reads the source output power from the IC6.
2. **Rate (Å/s)** Outputs the deposition rate read by the IC6.
3. **Gain (Å/s/%)** sets the deposition source gain.
4. **Set Gain** sends the gain that is entered to the IC6.
5. **MANUAL** toggles between Manual and Non-Deposit Control state.
6. **SHUTDOWN** shuts down the deposition source by switching the deposition source to Manual and setting the source output power to zero.
7. **Plot range (min)** sets the time window for which the rate/power plot should be displayed.
8. Plots the deposition rate (white) and the source output power (red).
9. **SHUTDOWN ALL** shuts off all deposition sources by switching all deposition sources to Manual and setting all source output powers to zero.
10. **CLOSE** quits the VI and returns to the main menu.

In order to get the source to the desired deposition rate, switch the source to the Manual state if necessary by pushing the MANUAL button (**5**). The source is in the Manual state if the button appears dark gray and pushed into the front panel (see Ti in the bottom left panel in Figure 3.2). In the Manual state, slowly increase the source output power (**1**) until a consistent non-zero rate appears (**2** and **8**).

⚠ Negative rates may appear occasionally due to communication problems with the IC6. If negative rates appear frequently, the crystal in the quartz crystal microbalance that measures the deposition rate may need to be replaced.

Keep increasing the power until the deposition rate is close to the desired deposition rate, and push the MANUAL button (**5**) to switch to Non-Deposit Control where the source output power and the deposition rate is controlled by the IC6. The button will now appear light gray and raised from the front panel (see Sn in the bottom right panel in Figure 3.2).

If the IC6 reacts very slowly to deviations from the desired deposition rate, or if the deposition rate oscillates very quickly and strongly, the gain needs to be adjusted. This can be done by entering the new gain value (**3**) and pushing the "Set Gain" button (**4**). A higher gain will decrease the power the IC6 uses to react to rate deviations. So, for strongly oscillating deposition rates, the gain needs to be decreased whereas for slowly oscillating rates, the gain needs to be increased.

In case of an emergency such as a short in the deposition source, the source can be shut down quickly by using the SHUTDOWN button (**6**). If all deposition sources need to be shut down, the SHUTDOWN ALL button (**9**) can be used.

⚠ The shutdown buttons only shut down the parts that are controlled by the IC6. Other hardware, such as the power sources for electron beam guns will still run after the shutdown buttons are pushed and need to be handled separately.

If the deposition source is not connected to the IC6, the deposition rate can still be monitored, but the source power output needs to be controlled manually from an external source. In Figure 3.2, Se (top right) is an example for such a source. Se is deposited using an effusion cell with a temperature controller that is not attached to the IC6, but the deposition rate is still output to the screen because the quartz crystal microbalance that monitors the deposition rate is attached to the IC6. The shutdown buttons will of course also be of no use with these sources. If the element is set to "None", all control and output elements for this element are disabled (top left in Figure 3.2).

Once all sources have the desired deposition rates, the VI can be closed using the CLOSE button (**10**), which will return the software to the main menu. From there, a new deposition can be set up and started.

3.2. Setting Up the Layering Sequence and Starting a Deposition

After starting up the deposition sources, the layering sequence of the deposition needs to be set up. A schematic layering sequence is shown in Figure 3.3. The white (w), black (b), and orange (o) colored bars represent different elements. A multi-layer deposition sequence consists of a repeating unit, which is deposited n times ($n = 4$ in Figure 3.3). The repeating unit in Figure 3.3 consists of the sequence b-w-b-w-o-w. To reduce the necessary user input, only one b-w layer will be configured and the other b-w layer is simply be repeated. These layers will be referred to as *layer sets*.

The overall repeating unit will thus be input as $\{[\text{b-w}]_2[\text{o-w}]_1\}$. Even though this example only uses two elements for each repeating layer set, there are no limits to how many elements can be used in a layer set or the repeating unit.

From the main menu, the “Deposition” button will lead to the sequence builder. Clicking on this button will write all elements and rates to the IC6, which is equivalent to pushing the “Update Elements” button. The sequence builder is shown in Figure 3.4 and consists of the following elements:

1. **Subunit panel**: The current layer set that is built or modified.
2. **Sequence**: The repeating unit.

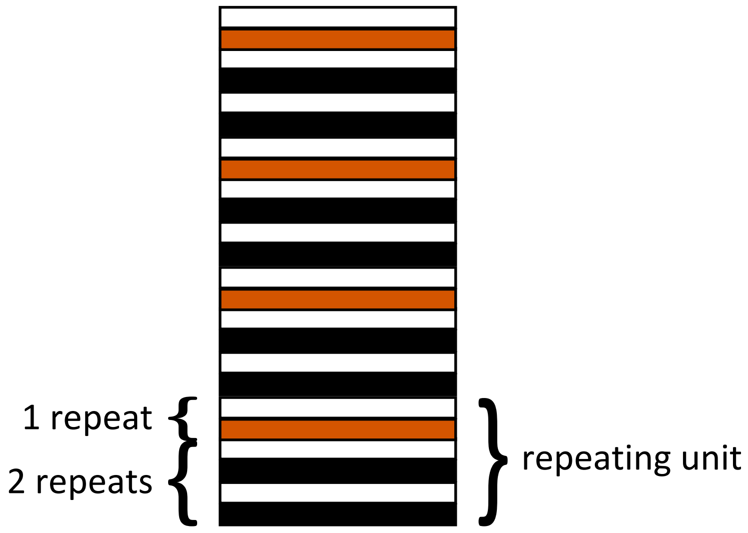
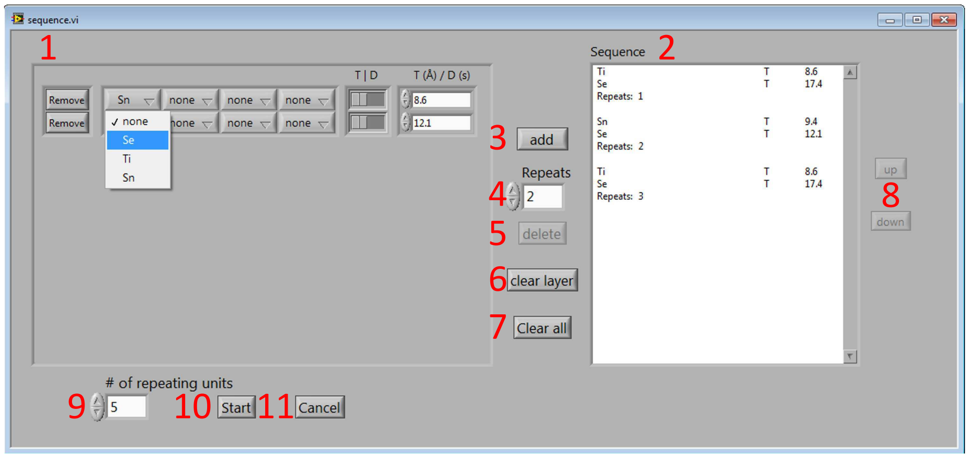


Figure 3.3: Schematic layering sequence of a deposition with white (w), black (b), and orange (o) “elements”. There are four repeating units, which consist of the sequence $\{[\text{b-w}]_2[\text{o-w}]_1\}$

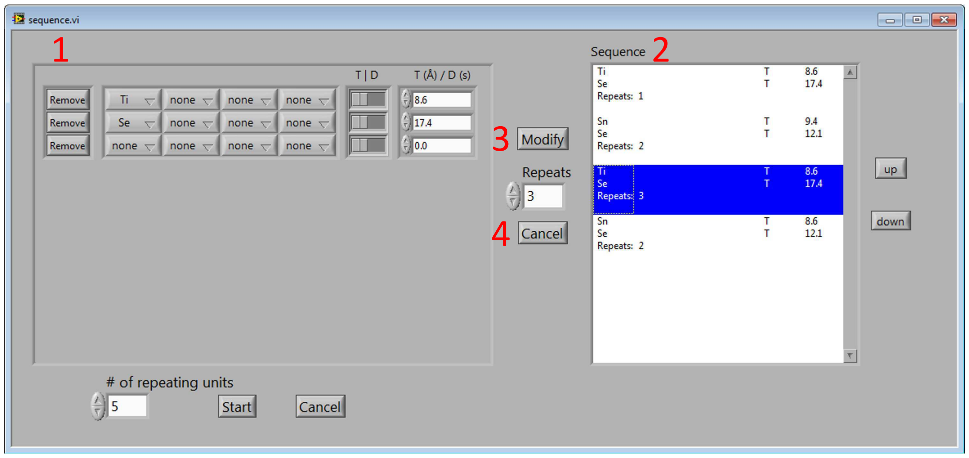


```
Ti          T    8.6
Se          T    17.4
Repeats: 1

Sn          T    9.4
Se          T    12.1
Repeats: 2

Ti          T    8.6
Se          T    17.4
Repeats: 3
```

Figure 3.4: Front panel of the sequence builder `sequence.vi`. For a description of the elements see text.

3. **Add**: Adds the layer set from (**1**) to the repeating unit.
4. **Repeats**: The number of layer sets from (**1**) that is added to the repeating unit.
5. **Delete**: Deletes a layer set that is selected in the repeating unit (**2**).
6. **Clear layer**: Clears the content in the layer set (**1**).
7. **Clear all**: Clears the content in the layer set (**1**) and the repeating unit (**2**).
8. **Up/Down**: Moves a layer set that is selected in the repeating unit (**2**) up/down.
9. **# of repeating units**: The number of repeating units.
10. **Start**: Starts the deposition.
11. **Cancel**: Exits the sequence builder and returns to the main menu.

The panel on the left (**1**) is used to build each layer set. The drop-down menus are used to select an element to be depoisted. Once an element is selected, a new row will appear automatically. The "T|D" switch toggles between two deposition modes: thickness mode (T) and deposition time mode (D). In thickness mode, the source shutter will close when the QCMs read the specified thickness. In deposition time, the shutter will close after a specified time. The text field on the right sets the thickness in Ångström and the time in seconds, respectively.



```
Ti          T   8.6
Se          T   17.4
Repeats: 1

Sn          T   9.4
Se          T   12.1
Repeats: 2

Ti          T   8.6
Se          T   17.4
Repeats: 3

Sn          T   8.6
Se          T   12.1
Repeats: 2
```

Layer set panel: Ti — 8.6; Se — 17.4; none — 0.0. Repeats: 3. # of repeating units: 5.

Figure 3.5: Front panel of the sequence builder when a layer set is modified.

Each row can be cleared using the “Remove” button on the left, or the entire layer set can be cleared with the “Clear layer” button (**6**). If a layer set is repeated, the “Repeats” parameter (**4**) needs to be changed as well.

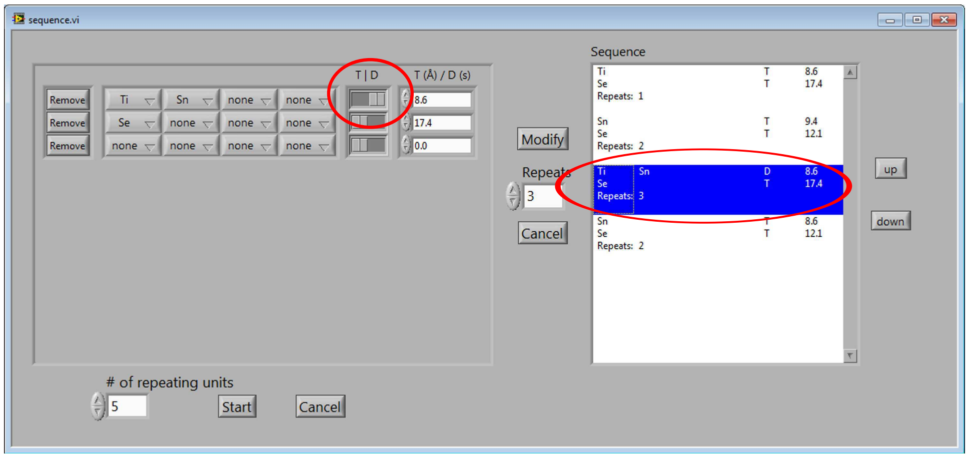
Once the layer set is built, it can be added to the sequence/repeating unit using the “Add” button (**3**). This will clear the layer set panel and a new layer set can be added to the sequence.

The sequence can be easily modified. After selecting a layer set with a single click, the layer set can be moved up and down (**8**) or completely deleted using the “delete” button (**5** in Figure 3.4).

The layer sets themselves can be modified by double-clicking on them. This will transfer the layer set to the layer set panel (**1**) and change the appearance of the front panel (see Figure 3.5). The layer set panel (**1**) can be manipulated the same way as when a layer set is added. The changes are saved with the “Modify” button (**3**), and cancelled using the “Cancel” button (**4**). The layer set panel will return to the layers it contained before the double-click.

The software also supports co-depositions, i.e. the deposition of more than one element at the same time. In order to co-deposit multiple elements, simply select an element in one of the unoccupied slots in a row as shown in Figure 3.6.

Co-depositions using thickness mode are currently not supported. If a second element is selected, the layer will automatically switch to deposition time mode.



				T | D	T (Å) / D (s)
Ti	Sn	none	none		8.6
Se	none	none	none		17.4
none	none	none	none		0.0

Repeats: 3

# of repeating units: 5

Sequence

```
Ti              T    8.6
Se              T    17.4
Repeats: 1

Sn              T    9.4
Se              T    12.1
Repeats: 2

Ti      Sn      D    8.6
Se              T    17.4
Repeats: 3

Sn              T    8.6
Se              T    12.1
Repeats: 2
```

Figure 3.6: Front panel of the sequence builder with co-deposited elements.

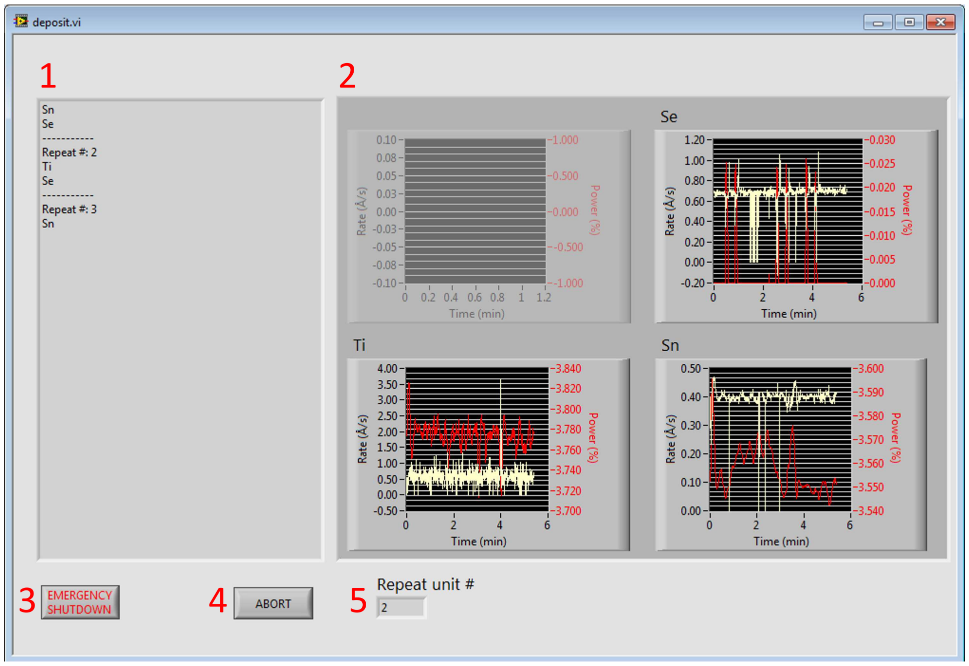
Finally, the number of repeat units needs to be set (**9**). The "Start" button (**10**) will start the deposition with the saved repeating unit. The "Cancel" button (**11**) will quit the sequence builder, and return to the main menu. The sequence will be saved in memory, so the changes are not lost when the sequence builder is cancelled or when a deposition is started.

⚠ The information in the layer set will not be used for the deposition. To deposit a layer set, it must be added to the sequence.

After clicking on the "Start" button, a new window will open and the deposition starts (Figure 3.7). The front panel of the VI monitors the progress of the deposition.

The panel on the left (**1**) displays the progress of the repeating unit that is being deposited. The last line in the panel is the current layer that is deposited. The panel clears with the beginning of a new repeating unit. The number of the current repeating unit is shown at the bottom of the front panel (**5**). The panel also shows the deposition rates and source power outputs of each element over the course of the deposition (**2**).

Depositions can be stopped before the last layer is deposited by either pushing the "Emergency Shutdown" (**3**) or the "Abort" button (**4**). The "Emergency Shutdown" button closes all shutters, sets all sources into the Manual state, and sets the source power outputs of all sources to zero. It is therefore suited for when there is a problem with the deposition source that require a shutdown of the deposition system. The "Abort" button also closes all source shutters, but leaves



```
Sn
Se
----------
Repeat #: 2
Ti
Se
----------
Repeat #: 3
Sn
```

Repeat unit # 2

Figure 3.7: Front panel of the deposition VI.

the deposition sources untouched. It is thus suited for when there is a mistake in the deposition setup or when the deposition rates become too unstable.

When the deposition ends, either because the last layer is deposited or because the deposition was aborted, a dialog will pop up asking the user if a new deposition should be started. This dialog is not available after an emergency shutdown.

If the user chooses “Yes”, the program will return to the sequence builder with the parameters of the last deposition, which can then be modified for a new deposition. If the user chooses “No”, the program will return to the main menu. The previous deposition is still saved in memory, so returning to the sequence builder from the main menu is equivalent to choosing “Yes” – that also means that accidentally clicking on “No” has no negative consequences.

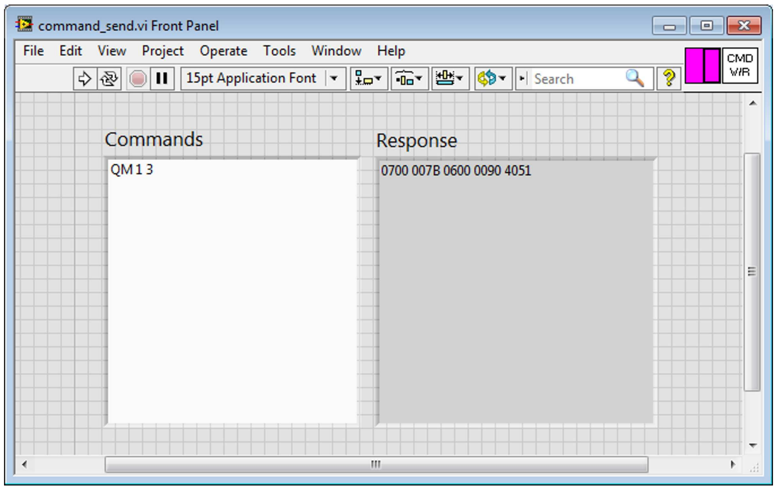
4. Communicating with the IC6

4.1. Sending Commands to the IC6

Occasionally, it may be necessary to directly communicate with the IC6, e.g. to troubleshoot the deposition system. The VI `command_send.vi`, which is extensively used by the software, can be used as a stand-alone program to send individual commands to the IC6. The VI `response_interpreter.vi` can then be used to convert the hexadecimal response into a readable format. This and the following section will provide an overview on how to use these VIs. For a complete command reference, see the IC6 manual and for a more detailed description of the IC6 message format, see section 5.2.2.

Commands need to be entered into the "Commands" text field (see Figure 4.1). Commands and each of their arguments need to be separated by a space character. Multiple commands can be sent at the same time by writting each command into a separate line. To send the command, simply run the VI by clicking the arrow in the toolbar. A response should appear in the "Response" text field.

In most cases, the VI automatically recognizes the data format of the input. Supported data formats are integers, float (decimal numbers), string (text), and logic statements. While the VI will assume that any format that is not recognized is a string, it is recommended to enclose strings with double quotes, especially when the string contains space characters. The VI does not recognize escaping – a backslash will always be interpreted as a backslash.


command_send.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
CMD WR
Commands
Response

```
QM 1 3
```

```
0700 007B 0600 0090 4051
```

Figure 4.1: Front panel of `command_send.vi` with the command `QM 1 3` (get density of element 3) and the response in hexadecimal.

There are two exceptions to the automatic data format recognition: 4-byte integers and logic statements. For the purpose of this program, it is necessary to differentiate between 1-byte integers and 4-byte integers. Integers that are entered into the "Commands" text field are assumed to be 1-byte integers. In order to send an integer as a 4-byte integer, the integer needs to be preceded by a # character. For example, the command `UM 38 1 2` requires 38 and 1 to be 1-byte integers and 2 to be a 4-byte integer. The command should then be entered as:

```
UM 38 1 #2
```

To determine whether an integer needs to be sent as a 1-byte or a 4-byte integer, see the command reference in the IC6 manual where 1-byte integers are referred to as "byte" and 4-byte integers are referred to as "integer".

The command to update a logic statement always starts with `UL YY if` where `YY` is the logic statement number. This is followed by the conditions for the logic statements. The conditions are finished up with `then`, followed by the actions for the logic statement. The actions are separated by `and`.

The logic conditions and actions are entered the way they are listed in the IC6 manual with two modifications: space characters need to be replaced by an underscore character, and XXX needs to be omitted. The following is an example for a logic statement command:

```
UL 1 if final_thick and not idle then go_to_postdep 1
```

4.2. Reading Responses

The response that `command_send.vi` outputs is the response it receives from the IC6. The response is encoded in hexadecimal and is described in detail in section 5.2.2. This section will describe how to quickly read and interpret responses and how to use the VI `response_interpreter.vi` to convert the hexadecimal responses into a readable format.

The response in the previous section as shown in Figure 4.1 will be used as an example. The response was:

0700 007B 06**00 0090 40**51

The response is split into sets of four characters. Two characters represent a hexadecimal number, so each response is split into sets of two numbers. The response shown above in decimal is:

7 0 0 123 6 **0 0 144 64** 81

If a command was sent and executed successfully, the third number (underlined) is always `00` and the fifth number (underlined) is always `06`. If these numbers are different, then the command was not sent or executed successfully – these error codes will be discussed in the following section.

The numbers behind `06`, except for the last number, all belong to the data that was requested (bold face). If multiple commands are sent to the IC6, the response will be sent in one single string, separated by `06`. For example, sending the commands `QM 1 3` and `QM 1 2` could result in a response like this:

0700 007B 06**00 0090 40**06 **3333 BB40** 51

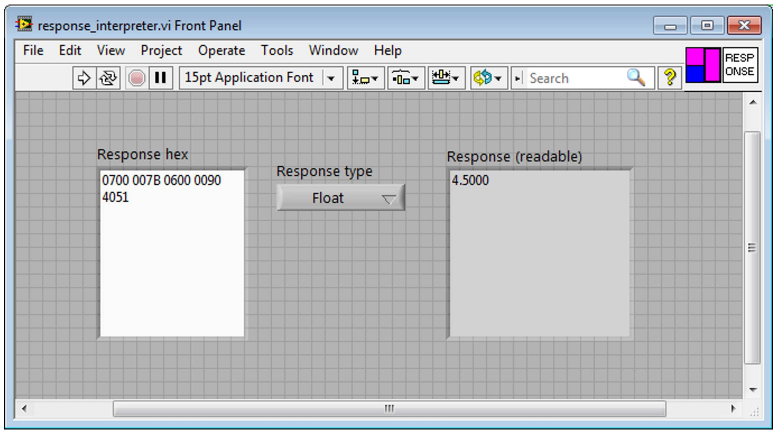
Where the bold face numbers belong to the responses to the individual commands. In some cases, the response does not have data, such as the commands `RL 6 1`, `RL 6 2`, etc., which set the deposition state to manual. Sending these commands for four sources at the same time would result in a response like this:

```
0600 005B 0606 0606 73
```

Since setting the deposition state to manual does not yield any data, there will be nothing past each occurrence of `06`.

The problem is that hexadecimal numbers can represent a multitude of data formats which are not readable by humans. However, the response can be converted into a readable format in the VI `response_interpreter.vi`. The front panel of this VI is shown in Figure 4.2.

To convert the response, the full response from `command_send.vi` needs to be pasted into the "Reponse hex" field, and the data type of the response needs to


response_interpreter.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
RESP ONSE

Response hex
0700 007B 0600 0090
4051

Response type
Float

Response (readable)
4.5000

Figure 4.2: Front panel of `response_interpreter.vi` with the response to the command `QM 1 3` converted to float.

be selected in the "Response type" drop-down menu. Supported response types are string, 4-byte integers (Integer), 8-byte integers (Long), float, 1-byte integers (Byte), and bits. To convert the response, simply run the VI by clicking on the arrow in the toolbar, and the response will be output in the text field "Response (readable)". In the case of the command `QM 1 3`, which asks for the density of material 3 (Ti), the response data type is float and the response is 4.5000 g/cm^3.

ⓘ The VI can only interpret the response of one single command. To interpret the response of a series of commands, each command has to be executed individually, or the response of each command needs to be entered separately enclosed by the first 10 and last 2 characters of the full response.

4.3. IC6 Error Codes

As mentioned in section 4.2, the third and fifth numbers of the response are `00` and `06` respectively, if a command string was sent and executed successfully. If these numbers are different, the command resulted in an error, and the values of these numbers represent the type of error returned by the IC6. These error codes are all listed in the IC6 manual, but will also be explained in this section to facilitate troubleshooting.

The third number can either assume the values `00` or `80`. If the value is `80`, the error is a so-called "packet error". A packet error is an error that appears when a packet, the message sent to the IC6, has the wrong structure. A list of errors with their error codes is shown in Table 4.1.

The following contains a brief description of each error and possible fixes:

Invalid checksum A checksum checks for the integrity of a packet. This error should not be raised under any circumstances unless changes were made to the VI `length_checksum.vi`. If this error occurs, revert to the original version of the VI.

Illegal format This error occurs when the command contains a wrong data type or is missing an argument. For example, the command `QM 1 3.5` would yield such an error because it expects two 1-byte integers as arguments and not a floating point number. A very common error would be to forget the # character in front of a 4-byte integer.

Table 4.1: Packet error codes when the third number in an IC6 response is `80`.

Error code (hex)	Error code (ASCII)	Description
43	C	Invalid checksum
46	F	Illegal format
49	I	Invalid message
4D	M	Too many commands
4F	O	Response length longer than response buffer

If all parameters are correct, the length byte could be wrong, which is only possible if changes have been made to the VI `length_checksum.vi`. Revert to the original version of the VI if that is the case.

Invalid message This error occurs when a command in the command string does not exist. For example, the command `QX 1 3` would raise such an error.

Too many commands The IC6 cannot take more than 100 commands per packet. If this error occurs, the command string needs to be split into multiple command strings with less than 100 commands.

Response length The response length is too long. If this error occurs, the command string needs to be split into multiple command strings to get shorter reponses.

If the third number is `00`, the IC6 returns a response code. A successful command will return `06`, which is the hexadecimal number of the ACK (acknowledgment) character. Other values point to an error where the command structure is correct, but the command itself can either not be executed or the command contains illegal elements. A list of reponse codes is displayed in Table 4.2.

The following contains a brief description of each response code and possible sources of error:

Acknowledgment The command was sent and interpreted successfully.

Illegal command The error occurs when a command is sent that the IC6 does not recognize. This error is similar to the packet error "Invalid message", and the difference between the two is only subtle. The difference is explained in more detail in section 5.2.2.

Illegal parameter value This error appears when a parameter has the correct format (1-byte integer, float, etc.), but has a value that is out of range. For

Table 4.2: Response codes when the third number in an IC6 response is `00`.

Response code (hex)	Response code (ASCII)	Description
06	ACK	Acknowledgment
41	A	Illegal command
42	B	Illegal parameter value
44	D	Illegal ID
45	E	Data not available
46	F	Cannot do now
4C	L	Length error
50	P	Prior command failed

example, `RL 6 227` would raise such an error because 227 is larger than the maximum value of 200.

Illegal ID The command ID, the first argument after a command, is invalid. For example, the command `H 5` would yield this response code because the Hello command (H) does not have the command ID 5.

Data not available As opposed to the error "Illegal parameter value", the parameter is in range, but there is no data written in the IC6. For example, if no material is written in layer/element 5, `RL 6 5` would yield this response code.

Cannot do now The IC6 is in the wrong state to execute the command. For example, executing the command `RL 6 1` (switch source 1 to the Manual state) while source 1 is already in the Manual state will raise this error. Refer to the IC6 manual to check in which state the IC6 has to be to execute the desired command.

Length error The command contains too many characters. The command must be 65,500 characters or less, and needs to be split into multiple commands if it contains more than 65,500 characters.

Prior command failed If multiple commands are sent to the IC6 at the same time, the IC6 will stop executing commands as soon as one command fails. For example, the command string `QM 1 3\\H 5\\H 1` would yield the response codes `06`, `44`, and `50`. The command `H 5` raises the "Illegal ID" error, which makes the command `H 1` fail with response code `50` even though it is a valid command.

5. Information for Developers

5.1. Overview of the VIs

This section provides a short overview of the structure of the program and the VIs it uses. The purpose of each VI and the arguments they take are also described in their respective block diagrams.

The program contains 12 VIs, 4 global variables, and 2 text files, which will all be explained in this section. The VI hierarchy shown in Figure 5.1 demonstrates how the VIs are connected.

The following is a brief description of each VI. For a more detailed description, see the corresponding block diagram. The VIs in alphabetical order are:

command_send.vi converts a list of commands into a format that can be read and interpeted by the IC6. The usage has been described in section 4.1, and the message format is outlined in section 5.2.2. The response is typically forwarded to `response_interpreter.vi` to convert it into an appropriate data type.

deposit.vi is the deposition engine and is mentioned in section 3.2. The deposition process is described in detail in section 5.3.

format_layer.vi is a helper VI for `sequence.vi`. It takes the layer that the user wants to add to the sequence or to modify and formats it by sorting the elements, removing duplicate elements in a row, and deleting empty rows.

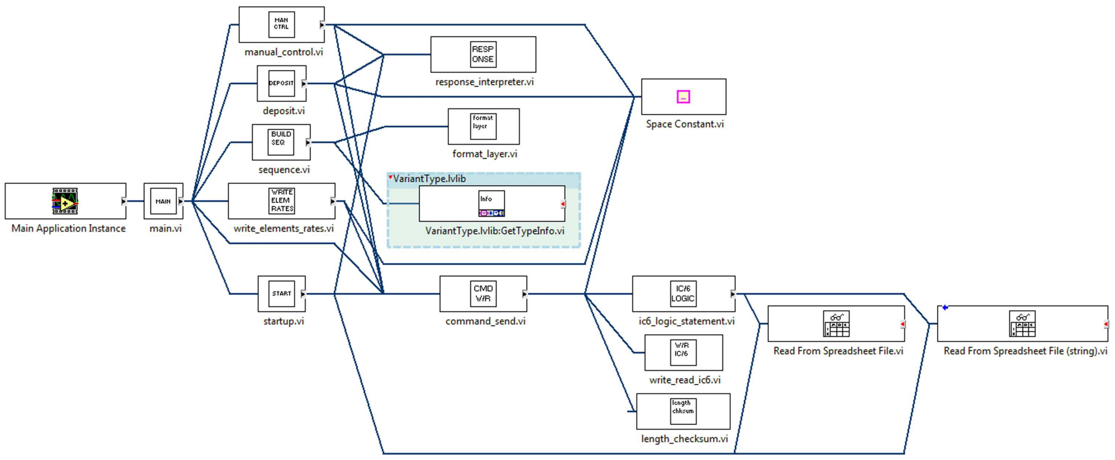


Figure 5.1: The VI hierarchy, including the LabVIEW libraries the program calls. The global variables are omitted for clarity.

The formatted layers are sent back to the listbox and the output cluster in `sequence.vi`

ic6_logic_statement.vi converts a logic statement to an IC6 command. This conversion is more complex than any other command, so it was moved into its own sub-VI and not implemented in `command_send.vi`. It uses a lookup table stored in the file `ic6_logic.txt`. The current version of the program does not use logic statements, but the VI was kept in case they are needed for troubleshooting or future expansions.

length_checksum.vi determines the length of the command sent to the IC6 and calculates the checksum. These numbers will be used by `command_send.vi` to complete the command that is sent to the IC6. For more information, see section 5.2.2.

main.vi is the main menu and the starting point of the program. It initializes the IC6 by leaving the STOP mode and setting the active process if required. It is briefly described in section 3.1.

manual_control.vi is used to start up the deposition sources as described in section 3.1.

response_interpreteri.vi converts the response from a single command from the IC6 into the indicated data type. It uses the response from `command_send.vi`, and also needs an integer input variable to designate the data type it converts the response into. The usage as a stand-alone program has been described in section 4.2.

sequence.vi is the sequence builder as outlined in section 3.2.

startup.vi sets system parameters and determines the current active process on the IC6. The parameters that are set are the active process number, the shutter timeout, and the number of sources (see the global variables list). This is also described in section 2.2. This VI also populates the global variable `elements.gbl` from the file `materialslist.txt` and creates the elements for the drop-down menu used in `main.vi`.

write_elements_rates.vi writes the selected materials and rates to the material slots on the IC6. It also sets a few behaviors of the IC6 that are necessary for the program to function properly in case the user manually overwrote these settings.

write_read_ic6.vi directly interfaces with the IC6. It takes the formatted command from `command_send.vi` and writes it to the IC6 buffer. It then reads the IC6 response buffer and returns the response to `command_send.vi`.

The global variables are mainly used to save global deposition parameters:

active_process.gbl is the variable for the designated active process.

elements.gbl contains the chemical formulas of the materials in the IC6 database. It is used in `sequence.vi` and `format_layer.vi` to populate drop-down menus, in `deposit.vi` and `manual_control.vi` for the rate and source power output plots, and in `deposit.vi` for the progress panel.

number_of_sources.gbl saves the number of sources in the deposition system. It is mainly used to extract the correct amount of data from the rate and source power output readings in `deposit.vi` and `manual_control.vi`.

shutter_timeout.gbl describes the time in milliseconds `deposition.vi` waits between each layer, i.e. it is the time between closing the source shutter of the current layer and opening the source of the next layer.

Finally, the program loads two text files:

ic6_logic.txt contains the IC6 logic elements and their corresponding hexadecimal value in a human-readable format. The sections in this file are just decorative to make the file easier to read for humans. The text file is used to create the lookup table in `ic6_logic.vi`.

materialslist.txt This file is a list of the materials stored in the IC6. It is used to populate the drop-down menus for the elements in `main.vi`.

5.2. IC6 Communication

5.2.1. Handling Data Types

Sections 4.1 and 4.2 described how to communicate directly with the IC6 using commands. The messages to the IC6 are not sent to the IC6 as plain text, but as bytes encoded as hexadecimal numbers. This requires a conversion between these bytes and different data types and vice versa. Commands use 1-byte integers, ASCII characters, single precision floating point numbers, and 4-byte integers.

1-byte integers can simply be sent as they are after converting the string to an unsigned byte, and ASCII characters can be transferred directly using their hexadecimal ASCII codes. Strings, however, have to be terminated by a NUL (`00`) character.

For larger integers or floats, the conversion is a little more involved. Integers are stored as signed integers and floating point numbers are stored as single precision floats, both of which are 4-bytes long. They can be converted into an array of bytes by converting the string into an integer/float and using the Type Cast function in LabVIEW. This array can be converted into a string of hexadecimal numbers. However, the IC6 requires the low byte to be sent first, i.e. the order of bytes needs to be reversed. For example, the number 4.52 as a single precision floating point number converts to the hexadecimal bytes `40 90 A3 D7` or `64 144 163 215` in decimal. This number needs to be sent as `D7 A3 90 40` to the IC6.

To convert responses, the hexadecimal numbers have to be converted into their intended data types. 1-byte integers can be directly converted into decimal, and ASCII characters can be converted using their hexadecimal values.

Integers longer than one byte and floating point numbers are returned with the low byte first, i.e. the number 4.52 will be returned as `D7 A3 90 40` and not as `40 90 A3 D7`, so the order of the bytes needs to be reversed before applying the Type Cast function.

There are two more data types in the responses the deposition program uses: 8-byte integers (Long) and bits. The former can be handled the same way as 4-byte integers, the latter can be converted using the Format Into String Function in LabVIEW.

5.2.2. IC6 Message Format

After successfully converting the commands into hexadecimal numbers, they need to be sent to the IC6 in the appropriate format. A schematic of the command format is shown in Figure 5.2.

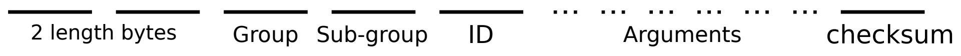


Figure 5.2: Format of an IC6 command. Each bar represents one byte.

The first two bytes are the length bytes, which are the number of bytes of the command with the low byte first. The last byte is the checksum, which is the sum of all bytes in the command modulo 256. The purpose of these three bytes is to check the integrity of the command packet, and generating these bytes is handled by `length_checksum.vi`.

Each command starts with a command group, which are sent as an ASCII character. The five command groups are listed in Table 5.3. Except for `H`, all command groups are accompanied by a command sub-group, which are listed in Table 5.4. The sub-groups are also sent as ASCII characters to the IC6. As the table shows, every available letter can accompany the command groups `Q` and `U`, but not `R` and `S`.

Combining a command group with an incorrect command sub-group will raise an "Illegal command" error (see section 4.3). If an ASCII character is used that is not in the tables, the packet error "Illegal format" is raised.

The following byte is always the command ID, which determines the type of command the IC6 is supposed to execute. The manual lists the meaning of all command IDs. If a command ID is used that is not assigned to a command, an "Illegal ID" error is raised.

A command may require additional arguments. The data types of the arguments depend on the command and need to be looked up in the IC6 manual. No special

Table 5.3: List of command groups and their corresponding symbols (ASCII).

Symbol	Command Group
H	Hello
Q	Query
R	Remote
S	Status
U	Update

Table 5.4: List of command sub-groups, their corresponding symbols (ASCII), and the command groups they can be combinded with. Note that L can correspond to "Logic" or "Layer", depending on which command group it accompanies.

Symbol	Sub-group	Command Groups
C	Source	Q, U
G	General	Q, R, S, U
I	Input Name	Q, U
K	Process Name	Q, U
L	Logic	Q, U
L	Layer	R, S
M	Material	Q, U
N	Material Name	Q, U
O	Output Name	Q, U
P	Process	Q, U
S	Sensor	Q, R, S, U
T	Type of Output	Q, U
V	User Message	Q, U

characters are needed to sperate multiple parameters, the IC6 will perform this separation automatically.

Multiple commands can be sent to the IC6 by concatenating the commands. No special characters are needed to separate commands. The only condition is that the commands are all between the length bytes and the checksum byte, which have to be determined for all commands together.

The format of the response that the IC6 returns to a command is shown in Figure 5.3. Just like the commands, the response starts with two length bytes and ends with a checksum byte. The bytes after the length bytes are the Condition Code Byte (CCB), a timer byte and either a package error code or a response code, depending on the value of the CCB.

The CCB displays whether the command packet the IC6 received was valid or not. The IC6 only writes the most significant bit (MSB), i.e. the bit with the highest value, of the CCB byte. It can thus only take the values `00000000` (0 in decimal and hexadecimal) or `10000000` (128 in decimal and 80 in hexadecimal). If the MSB is set to 1, i.e. the CCB is `80` in hexadecimal, there is an error in the packet – if it is set to `00`, the packet is valid.

The timer byte is simply a running integer number modulo 256. This integer is increased every 0.1 seconds.

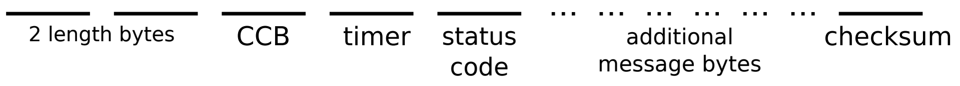


Figure 5.3: Format of an IC6 response. Each bar represents one byte. The status code is either a package error code or a response code.

The status code byte either represents a packet error code if the CCB is `80`, or a response code if the CCB is `00`. The meaning of these codes are shown in Tables 4.1 and 4.2 in section 4.3. A command was successfully sent and executed when the CCB is `00` and the response code is `06`.

After these three bytes, the IC6 sends the response byte(s) if any. When multiple commands are sent, each command has its own response code followed by the response message bytes. If there is an error in one of the commands, there will be no message bytes after the command that caused the error as shown in section 4.3. If the error is a packet error, there is only one packet error code, regardless of how many commands were sent, and if any of them could have been executed.

5.3. The Deposition Process

In order to deposit a custom layering sequence with an arbitrary number of layers, a deposition algorithm has to meet the following requirements:

- It must be able to deposit the layering sequences outlined in Figure 3.3.
- Any deposition sequence must be possible without limitations.
- The deposition rates have to be kept constant throughout the entire deposition.
- For thin layers, each layer has to be deposited with a precision of at least 0.1 Å.

The IC6 has its own programming interface that can be used with the front panel. Materials can be assigned to sources with a set deposition rate. These can then be assigned to layers in a process, which sets up the layering sequence. A process can contain up to 200 layers, and up to 50 processes are available. Up to six layers can be shown at a time on the front panel during the deposition.

For a multi-layer deposition with thin layers, however, this is not suitable for the following reasons:

- A deposition with four layers in a repeating unit and 40 repeating units, which is standard for ferecrystal precursors of 50 nm thickness, needs 160 layers. Thicker films rapidly approach the process limit, so a new process needs to be defined and executed right after the first process is finished.
- Only two materials can be co-deposited with a controlled rate. Each has to be in a separate layer, which further inflates the number of layers in the process.
- All the information needs to be typed into the front panel, which is not very convenient. A repeating unit will have to be entered, copied and then pasted for example 40 times. This is especially cumbersome when a deposition parameter for one material needs to be changed.
- The front panel displays a maximum of six layers (not materials), so if a material is not deposited within the next six layers, the deposition rate and the status of the deposition source cannot be monitored.
- The final thickness can only be entered with a precision of 0.001 kÅ, i.e. 1 Å. Sub-Å precision is not possible with the front panel.

To solve these problems, this program uses the following workarounds:

- The sources are assigned to layers of the same number. For example, when titanium is in source 3, it will be assigned to source 3 at the IC6, and in turn, source 3 will be assigned to layer 3. That way, the sources can be monitored on the front panel.

- Layers will be deposited by dynamically setting the layers into Deposit mode. The behavior on final thickness will be set to NonDepCtrl (Non-Deposit Control) so that the deposition rate is kept constant when the source shutter closes. That way, the maximum number of layers is unlimited, and any number of materials can be co-deposited.

- The layering sequence in Figure 3.3 and the use of repeating units, sets of layers, and repeats minimizes the user input and the amount of effort needed to change only one parameter for each layer.

- To achieve 0.1 Å precision, the final thickness value is sent to the IC6 with a higher precision than the front panel allows.[5]

The VI `write_elements_rates.vi` partially implements the first two points by assigning the materials to the respective sources, but also by defining the actions on final thickness. The VI configures the IC6 to go to NonDepCtrl on final thickness so that the source shutter closes and the deposition rate is kept constant.

The materials are assigned to their respective layers in `main.vi` on startup if they are not assigned correctly already. This is done automatically if the active process number needs to be changed.

The remaining points are implemented in the deposition algorithm that is executed by `deposit.vi` based on the layering sequence built in `sequence.vi`. The latter outputs two important variables, the Input Array, which contains the information for the repeating unit, and an integer for the number of repeating units, both of which are used in the deposition algorithm.

The structure of the Input Array is shown in Figure 5.4 and mimics the layering structure shown in Figure 3.3. Each array element is a cluster that represents one set of layers. These clusters contain three arrays and an integer.

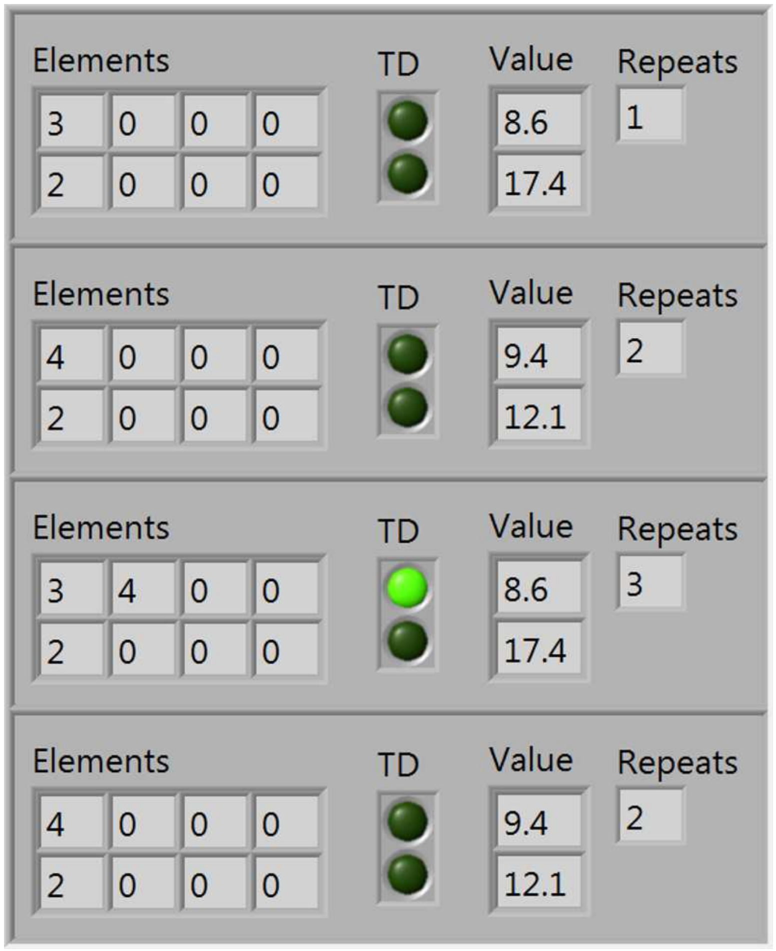
The interger ("Repeats") is the number of repeats, i.e. the number of times the set of layers is repeated. The three arrays are the arrays that are used to set up the sequence of layers in `sequence.vi`.

[5]The manual also only mentions a precision of 1 Å, but the IC6 does accept more precise values. This can be tested by measuring the time the source shutter is open. Using a deposition rate of 0.3 Å/s and depositing in increments of 0.3 Å yields a difference of exacly 1 second.

The “Elements” array is a 2D array that contains one row for each deposited layer, and each row contains the materials in the corresponding layer. However, the materials are not encoded by the element name, but by the deposition source number so that they can easily be addressed in the IC6 where they are stored as layers. An array element that is zero will be ignored by the deposition process.

The array “TD” is a Boolean array that indicates whether a layer is deposited in thickness mode (T) or deposition time mode (D). For thickness mode, the array element is set to false, for deposition time mode, the array element is set to true.

The “Value” array contains either the thickness in Ångström or the deposition time in seconds, depending on whether the value in the “TD” array with the same index is true or false.



Elements				TD	Value	Repeats
3	0	0	0	●	8.6	1
2	0	0	0	●	17.4	

Elements				TD	Value	Repeats
4	0	0	0	●	9.4	2
2	0	0	0	●	12.1	

Elements				TD	Value	Repeats
3	4	0	0	🟢	8.6	3
2	0	0	0	●	17.4	

Elements				TD	Value	Repeats
4	0	0	0	●	9.4	2
2	0	0	0	●	12.1	

Figure 5.4: Structure of the input array for the deposition. The example is taken from the deposition sequence shown in Figure 3.6

This structure is reflected by the deposition algorithm that is executed by `deposit.vi` as shown in Algorithm 1. There are four for loops in this algorithm. The outermost loop represents the repeating unit. One level below, the loop goes over each set of layers, i.e. it loops over each element of the Input Array.

Inside this loop, the cluster is unbundled and passed into the next loop, which represents the number of repeats of this layer set. At last, the fourth loop runs over all layers inside a layer set. Each instance of this loop starts the actual deposition procedure, which will be outlined in Algorithm 2. The elements of each layer is output into the corresponding panel in `deposit.vi`.

Once the set of layers with all its repeats is deposited, a decorative divider is added to the front panel, and the next set of layers is started. The entire procedure is repeated as many times as there are repeating units. After finishing all for loops, the user is asked if another deposition should be started, and the VI will output the response to `main.vi`.

All for loops can be determined prematurely by pushing the "Abort" or "Emergency" button. This will terminate all loops and move to line 23 in Algorithm 1. All sources on Deposit will be set to NonDepCtrl to close the source shutter. If the "Emergency" button was pushed, all sources will be set to Manual afterwards, and the source power outputs will be set to zero. The VI returns false automatically in this case whereas for "Abort", the user can decide.

The deposit procedure that is called by Algorithm 1 is shown in Algorithm 2. It starts with a check if the layer is to be deposited in thickness (TD == false) or deposition time mode (TD ==true). In the block diagram of `deposit.vi`, this is the innermost case structure.

In thickness mode, only the first element is taken because the other elements are zero. The final thickness of this element/layer is sent to the IC6 in the first step. Since the final thickness needs to be sent to the IC6 in kÅ, the value needs to be divided by 1000.

⚠ The value needs to be converted into a string before it is sent to `command_send.vi`. In the current version, the precision is set to 4 (0.0001 kÅ or 0.1 Å). If a higher (or lower) precision is desired, the value needs to be changed accordingly. Changing the precision in `sequence.vi` is not enough!

The layer will then be set to Deposit, and the source shutter will open until the final thickness is achieved, after which it will go to NonDepCtrl and close the shutter while keeping the rate constant. While the source shutter is open, the VI will continuously determine the state of the layer, the abort button, and the

Algorithm 1: Pseudocode of the procedure that sets up the layers to deposit in `deposit.vi`. Indexing starts with 1. The Boolean variables "abort" and "emergency" denote if the "Abort" or "Emergency" button was pushed. The pseudocode for the procedure "deposit" is shown in Algorithm 2.

Input: repeating_units, input_array from `sequence.vi`
Output: start_new

```
1  for unit = 1 to repeating_units do
2      sets = size(input_array);
3      for s = 1 to sets do
4          elements = input_array[s]['Elements']
5          TD = input_array[s]['TD'];
6          values = input_array[s]['Values'];
7          repeats = input_array[s]['Repeats'];
8          for r = 1 to repeats do
9              if $repeats > 1$ then
10                 print('Repeat #%s' % r);
11             layers = size(TD);
12             for l = 1 to layers do
13                 if abort or emergency then
14                     goto line 23;
15                 else
16                     call deposit(elements[l], TD[l], values[l]);
17                 end if
18             end for
19         end for
20         print('——');
21     end for
22 end for
23 if abort or emergency then
24     for source in to sources do
25         if Deposit then
26             set to NonDepCtrl;
27         end if
28     end for
29 end if
30 if emergency then
31     Set all sources to Manual;
32     Set all source power outputs to zero;
33     start_new = false;
34 else
35     print('Start another deposition?');
36     read(start_new);
37 end if
38 return start_new
```

emergency button. It will stop the loop if either of these buttons are pushed, or if the layer is in the NonDepCtrl state.

The purpose of this while loop is to prevent the VI from moving to the next layer, and does not affect the deposition itself. After the shutter closes, the VI will wait before the next layer is deposited. The waiting time is determined by the "Shutter timeout" variable that is set in `startup.vi`. After this timeout, the VI will move on to the next layer.

In deposition time mode, the final thickness of all elements in the layer is set to 999.99 kÅ, the highest possible value, to prevent the source shutter from closing prematurely. The elements are then all set to Deposit, and the VI enters a while loop that checks if the desired time has passed, or the abort button or the emergency button was pushed. Afterwards, the elements are set to NonDepCtrl, and the VI waits before the next layer is deposited.

Algorithm 2: Pseudocode of the deposit procedure that is called in Algorithm 1. shutter_timeout is the global variable that was set in `startup.vi`. The Boolean variables "abort" and "emergency" denote if the "Abort" or "Emergency" button was pushed.

```
Input: elements, TD, value
if thickness mode then                                /* TD == false */
    element = elements[0];
    print(element symbol);
    $t$ = value/1000;                      /* IC6 takes thickness in kÅ */
    Set final thickness of element at IC6 to $t$;
    Set element to Deposit mode;
    while not NonDepCtrl and not abort and not emergency do
        wait 250 ms;
        check status of element at IC6;
    end while
else
    for element in elements do
        if element $\neq$ 0 then
            print(element symbol);
            Set final thickness of element to 999.99 kÅ;
            Set element to Deposit mode;
        end if
    end for
    $t_0$ = current time;
    $t_1$ = current time;
    while $t_1 - t_0 <$ value and not abort and not emergency do
        wait 100 ms;
        $t_1$ = current time;
    end while
    for element in elements do
        Set elements to NonDepCtrl;
    end for
end if
wait shutter_timeout ms;
```