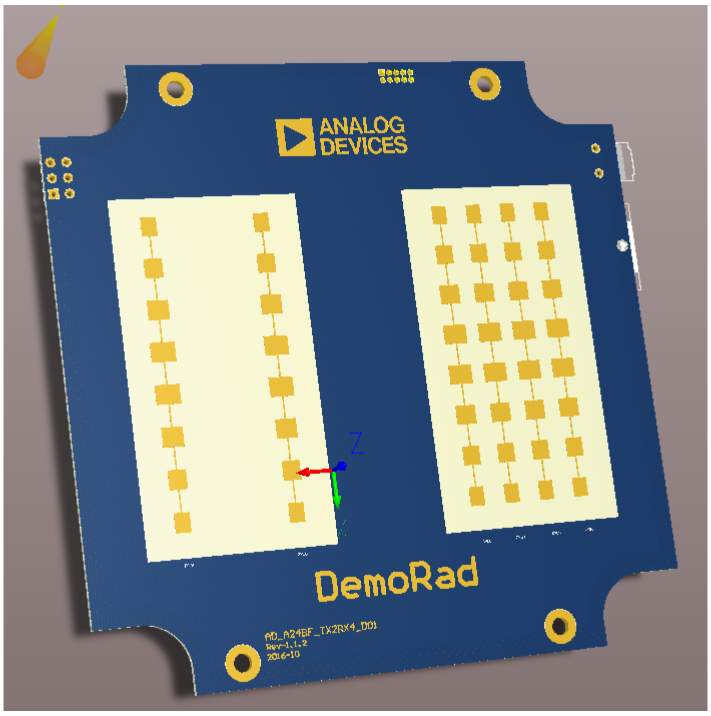
DemoRad

(GUI Specification)



Inras GmbH
Altenbergerstrasse 69
4040 Linz, Austria
Email: office@inras.at
Tel. Nr.: +43 732 2468 6384

Linz, 2015-04-24

Contents

1 Document Version **2**

2 Graphical User Interface **3**
2.1 USB Interface 3
2.1.1 USB Command Structure 4

3 Integration in Python **5**

4 Typical Processing Flow for Operating the Radarsystem **5**

5 Supported Commands **7**
5.1 Cmd: DspEnTstDat 7
5.2 Cmd: DspSetADICfg 7
5.3 Cmd: DspWrCalDat 7
5.4 Cmd: DspGetSts 8
5.5 Cmd: DspGetChirp 8
5.6 Cmd: GetVersInf 8
5.7 Cmd: DspSetSpi 9

1 Document Version

Version	Description	Date	Author
1.0.0	Initial Version	2015-04-24	Andreas Haderer

2 Graphical User Interface

The graphical user interface (GUI) supports the following measurement modes

- FMCW measurement view,
- Range-Doppler view, and
- MIMO processing.

The GUI uses a USB 2.0 connection to communicate with the DSP. In the following section the suggested USB command structure is explained in more detail. The commands are used to configure the signal processing chain, the data interface, and the 24-GHz FMCW frontend. In addition commands for reading the DSP status information are required.

2.1 USB Interface

In Fig. 1 a sketch of the USB communication structure with the PC being the master and the DSP being the client is shown.

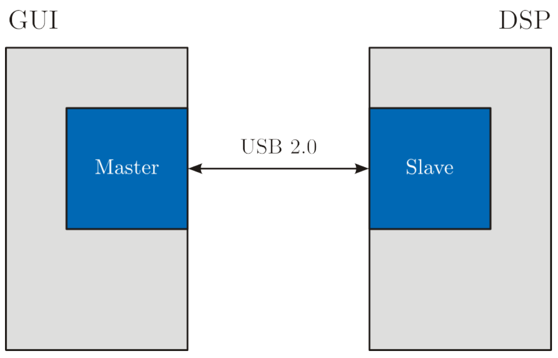


Figure 1: USB communication between PC and DSP.

For integrating the radar system in the GUI the connection must support read and write operations. In the following section the structure of the configuration commands are explained in more detail.

2.1.1 USB Command Structure

In the following section the proposed command structure is explained in more detail. The defined structure is a first proposal and changes to the data types, the number of words, and the word size can be implemented, if it simplifies the implementation on the DSP side. But for a responsive GUI it is essential that a communication in both directions is possible during operating the radar system.

Every command consists of a 32-Bit header and a fixed number of data words (32-Bit) as shown in Tab. 2.1.1. The length of the used data words is encoded in the header and in addition the header contains a unique command code, which is used to identify the command. The N data words (32-Bit) contain an arbitrary payload. The meaning of the payload depends on the command code.

Header	32-Bit with length and command code
Data word 1	32-Bit with payload data 1
Data word 2	32-Bit with payload data 2
...	
Data word N	32-Bit with payload data N
...	
Data word 30	unused data word
Data word 31	unused data word

Table 1: Configuration command structure with N < 32 and a fixed size of 32 words.

The definition of the header is summarized in Tab. 2 to Tab. 5.

The RD flag can be used to enable the response from the slave if a command requires to read back information form the DSP. The fields $L4-L0$ define the length (number of 32-Bit words) of the command, where the header is included in the command length. The shortest command consists only of a header and therefore has a payload length of $L = 1$. The longest command consists of 32 words (32-Bit), where 31 words are used for the payload. The command code is defined by the fields C15-C0. The command code is used to decide how the payload is interpreted and which data is returned in case of RD is set to 1.

H31	H30	H29	H28	H27	H26	H25	H24
RD	-	-	-	-	-	-	-

Table 2: Byte 3 of 32-Bit header.

-	-	-	-	-	-	-	-
-	-	-	L4	L3	L2	L1	L0

Table 3: Byte 2 of 32-Bit header.

The required command codes are defined in Sec. 5.

H15	H14	H13	H12	H11	H10	H09	H08
C15	C14	C13	C12	C11	C10	C09	C08

Table 4: Byte 1 of 32-Bit header.

H07	H06	H05	H04	H03	H02	H01	H00
C07	C06	C05	C04	C03	C02	C01	C00

Table 5: Byte 0 of 32-Bit header.

3 Integration in Python

To integrate this communication in Matlab or Python, a driver is required, with the following functions

- Open device,
- Write data to device (write array),
- Read data from device (read array), and
- Close device.

For integrating the driver in Python or Matlab, a C++ implementation of the driver would be an advantage. In this case a Mex file for Matlab can be generated, so that an operation of the board is also possible with a Matlab script.

4 Typical Processing Flow for Operating the Radarsystem

In this section we describe a typical processing flow, which is required to operate the radar system from the Inras GUI. In Fig. 2 the command flow for starting the measurements is shown. In the first stage the GUI reads the software version of the DSP code. This is required to maintain a single GUI for different DSP software versions. Depending on the software version the GUI can adjust the configuration flow and or reduce the set of provided functions.

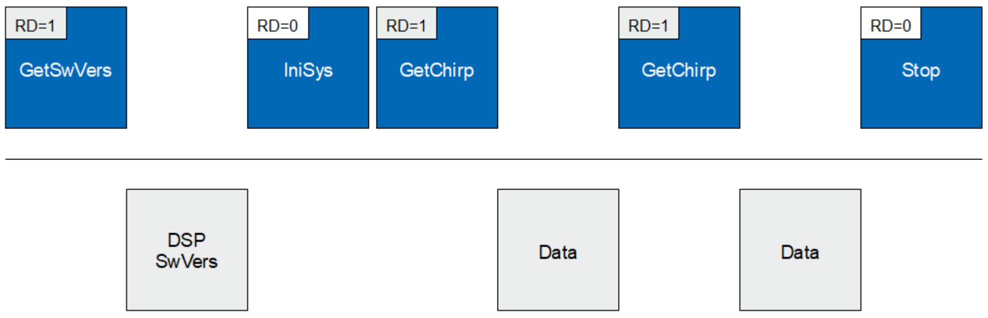


Figure 2: Typical command flow for a measurement cycle with initialization.

After reading a valid software version information, the GUI initializes the system. The initialization step will include multiple commands for initializing the different components on the radar system. The initialization of the RF components must enable a direct write to the registers of the components. Therefore commands for reading and writing registers to the ADF5901, ADF5904, ADF4159, and the ADAR7251 must be implemented. In addition, commands for initializing the DSP chain, including the desired measurement timing will be required. After the initialization phase is finished, the GetChirp command is executed by the GUI. With this command measurements are started according to the configured timing mode. At the same time the GUI starts to read back the measurement data and the DSP return the data if a new set of signals is available. This is repeated until the user stops the measurements. In this case the stop command is transmitted from the GUI to the DSP.

5 Supported Commands

In this section the supported commands are summarized. In Tab. 6 a list of the available commands is summarized.

Command	Code	Description	Ref
DspEnTstDat	0x6005h	Enable generated Test Data	Sec. 5.1
DspSetADICfg	0x6006h	Config Board with ADI structure	Sec. 5.2
DspWrCalDat	0x6007h	Write Calibration Data to Flash	Sec. 5.3
DspGetSts	0x6008h	Get DSP Board Status	Sec. 5.4
DspGetChirp	0x7003h	Get Chirp Data	Sec. 5.5
GetVersInf	0x900Eh	Get System Version Info	Sec. 5.6
DspSetSpi	0x9017h	Sends SPI commands from the DSP	Sec. 5.7

Table 6: Available commands.

5.1 Cmd: DspEnTstDat

The DspEnTstDat command enables/disables testdata, which is returned with DspGetChirp instead of the measured data.

Word	Description
0	Header
1	if 1 : Enable, else : Disable

Table 7: Values for DspEnTstDat command.

5.2 Cmd: DspSetADICfg

The DspSetADICfg command enables the usage of an ADI configuration values array. It is necessary to first send the DspSetADICfg command for ADI configuration, which puts the board into a mode, where the data can be read as an array. The configuration input type is selected via the config selection word, which is described in detail in Tab. 9.

Word	Description
0	Header
1	Config selection

Table 8: Values for DspSetADICfg command.

5.3 Cmd: DspWrCalDat

The SetCal command is used to store calibration data to a non-volatile memory on the DSP board. It is necessary to first send the DspWrCalDat command for ADI calibration data, which puts the board into a mode, where the data can be read as an array.

Word	Description
0	Set Adf5901
1	Set Adf5904
2	Set Adf4159
3	Set Adar7251
4	Set ModFmcw

Table 9: Values for Config selection field.

Word	Description
0	Header
1	Calibration selection
2	if Inras Cal Dat: CalData[0]
...	
Len	CalData[Len -1]

Table 10: Values for DspWrCalDat data command.

Word	Description
0	Set ADI Cal Data
1	Get ADI Cal Data
2	Set Inras Cal Data
3	Get Inras Cal Data

Table 11: Values for Calibration selection field.

5.4 Cmd: DspGetSts

The DspGetSts command always returns 1 on every query and is used to check if the board is connected and running.

Word	Description
0	Header
1	Status

Table 12: Return values for DspGetSts command.

5.5 Cmd: DspGetChirp

The DspGetChirp command returns the sampled chirps via the USB connection. After the DspGetChirp command was sent, the board returns the data in an array. The format of the returned array depends on the settings of the ADAR7251 SPI and DMA.

5.6 Cmd: GetVersInf

The GetVersInf returns the version information of the board.

Word	Description
0	Header
1	Chirp start
2	Chrip stop

Table 13: DspGetChirp command definition.

Word	Description
0	Header
1	System Software Variant
2	System ID
3	Hardware ID

Table 14: GetVersInf command definition.

5.7 Cmd: DspSetSpi

The DspSetSpi command sends data via SPI on the specified channel directly to the specified device.

Word	Description
0	Header
1	Place holder
2	SPI Channel selection
3	SPI Command Data[0]
...	
Len	SPI Channel[Len -1]

Table 15: DspSetSpi command definition.

Word	Description
0	ADF5904
1	ADF5901
2	ADF4159
3	ADAR7251

Table 16: SPI Channel selection definition.