4.7 FreePort communications

4.7.1 Simple download of data to the target

The target model *FreePortComm_RX_simple.mdl* and the corresponding host model *FreePortComm_TX_simple.mdl* demonstrate the use of the *FreePort* communication blocks for data download from the host to the target. The *FreePort* communication interface does not rely on the External Mode and can therefore be used with small standalone programs as well as in parallel with optional External Mode communications via the other port. The sample target model receives 5 values on its SCI0 port and feeds these values to an output block for PTH. The latter block has been configured to switch a line high whenever the incoming signal exceeds 3.5. A port line is switched low when the input falls below 2. Port SCI0 is the free communication port on the Dragon-12 when using External Mode. Without External Mode communications, both ports are free. The display shown in Figure 4-11 only shows the incoming data when using External Mode communications.

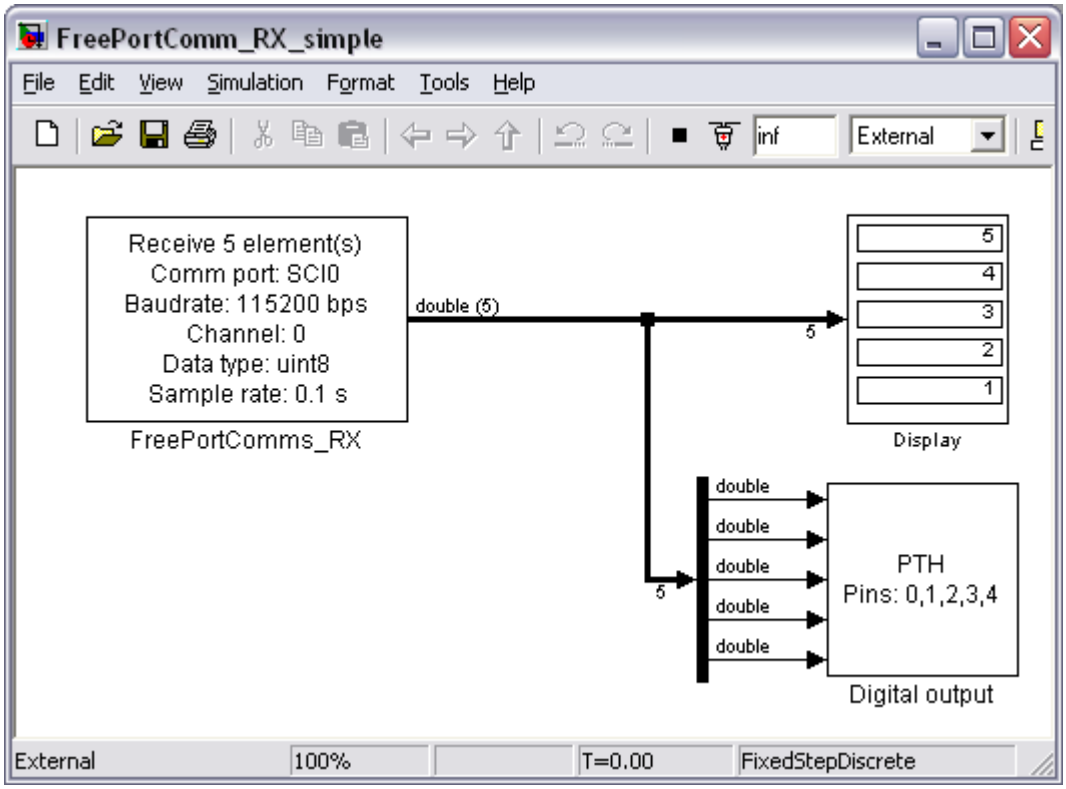

Figure 4-11 Sample model: FreePortComm_RX_simple.mdl (target)

Figure 4-12 is the corresponding host sided block diagram. Note that, instead of using a host-sided Simulink model, the command *freePortSend* could have been used. This is often more convenient, especially when the data to be sent is produced by an m-file.

Command *freePortSend* has the following syntax:

freePortSend(1, 115200, 0, 5, 2, [1 2 3 4 5])

This sends data via COM1, at 115200 bps using channel 0. The number of elements to be sent is 5 and they are of type 2 (uint8; 0 = single, 1 = int8, 2 = uint8, 3 = int16, 4 = uint16, 5 = int32, 6 = uint32, 7 = boolean); the data values are then numbers 1 – 5. For an example of how to use command *freePortSend* see the test m-file script *freeport_test.m*.

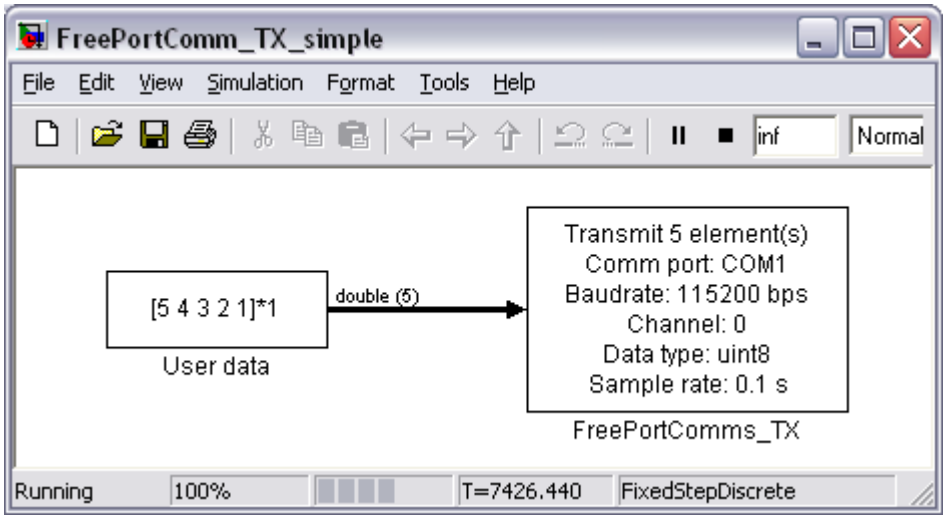

Figure 4-12 Sample model: FreePortComm_TX_simple.mdl (host)

4.7.2 Simple upload of data from the target

The target model *FreePortComm_TX_simple2.mdl* and the corresponding host model *FreePortComm_RX_simple2.mdl* demonstrate the use of the *FreePort* communication blocks for data upload from the target to the host. The target model reads the dip switches connected to port H (PTH) and uploads this information (‘0’ or ‘1’) to the host. The optional (target sided) display block is only serviced when running in External Mode (Figure 4-13).

Figure 4-14 shows the corresponding host model. Note that, instead of using a host-sided Simulink model, the command *freePortReceive* could have been used. This is often more convenient, especially when the received data is to be processed further by an m-file.

Command *freePortReceive* has the following syntax:

[myData, numElementsReceived] = freePortReceive(1, 115200, 0, 5, 2, 1)

This attempts to receive data via COM1, at 115200 bps using channel 0. The number of elements to be received is 5 and they are of type 2 (uint8; 0 = single, 1 = int8, 2 = uint8, 3 = int16, 4 = uint16, 5 = int32, 6 = uint32, 7 = boolean). A blocking call is made to *freePortReceive* (last call-up parameter, 1 = blocking, 0 = non-blocking), i. e. the command only returns, once some data has been received – if the received data is not

destined for the chosen channel (here: '0'), both return parameters are '0'; when the received data is destined for the chosen channel, *numElementsReceived* should match the expected number of elements (here: '5') and *myData* are the received data values. The latter are of the expected type (here: 'single'). For examples of how to use command *freePortReceive* see the test m-file scripts *freeport_test2.m*, *freeport_test3.m* and *freeport_test4.m*.

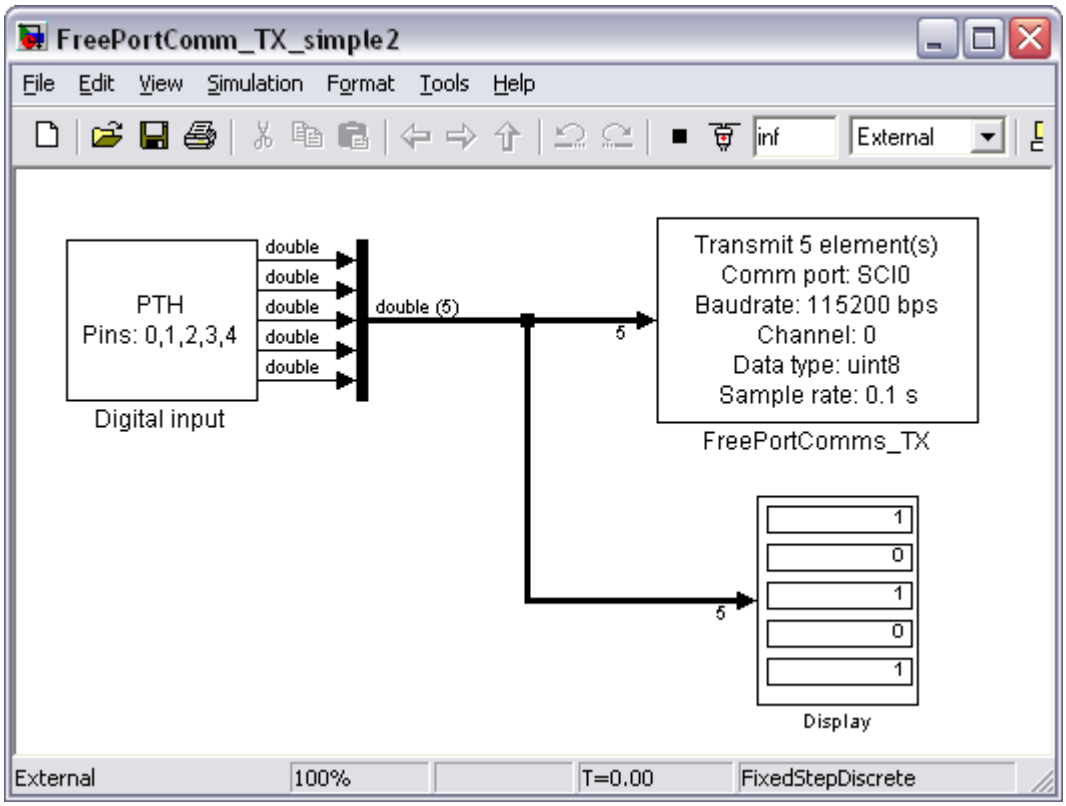

Figure 4-13 Sample model: FreePortComm_RX_simple2.mdl (target)

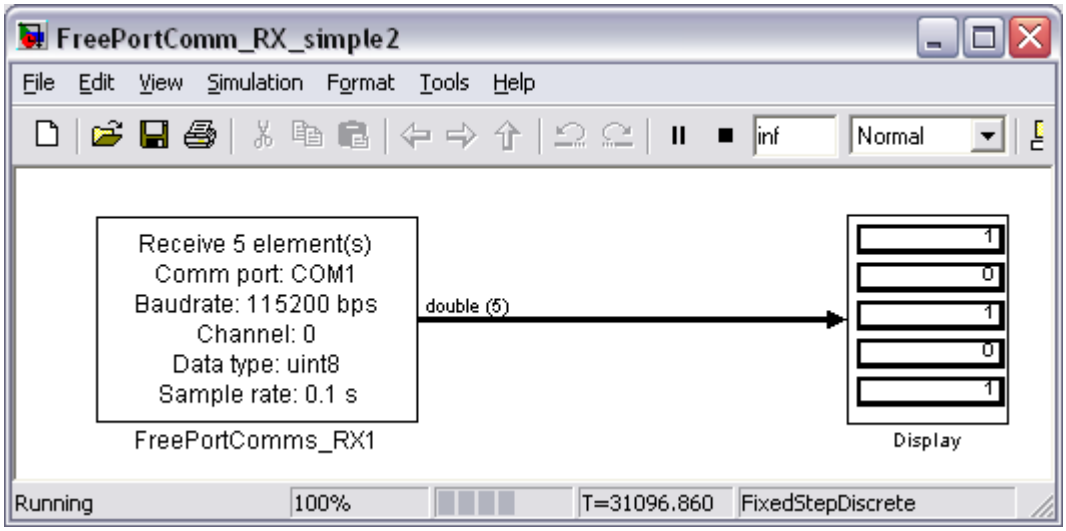

Figure 4-14 Sample model: FreePortComm_TX_simple2.mdl (target)

4.7.3 Simultaneous upload and download of data between host and target

The target model *FreePortComm_RXTX.mdl* and the corresponding host model *FreePortComm_TXRX.mdl* demonstrate the simultaneous upload and download between host and target using the FreePort communication blocks. Figure 4-15 is the target-sided model, whereas Figure 4-16 shows the host-sided equivalent.

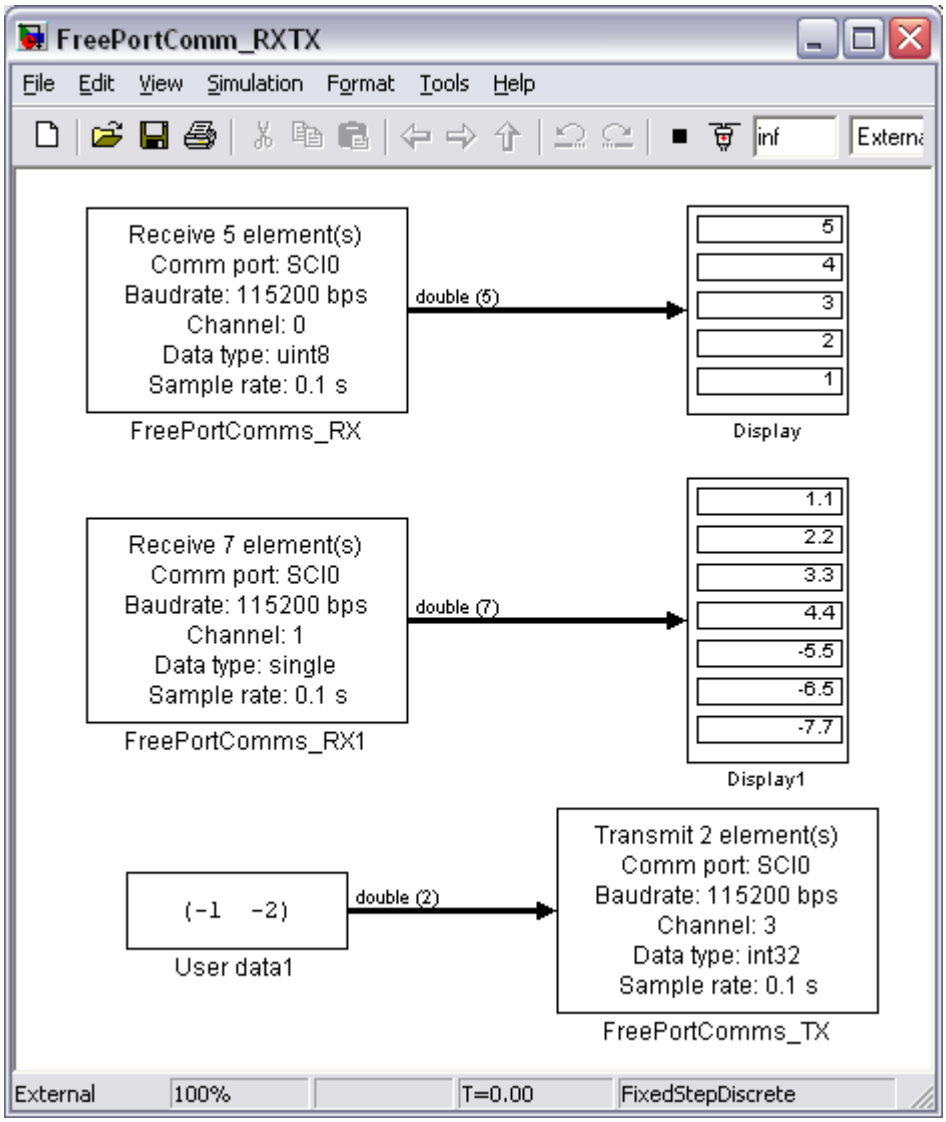

Figure 4-15 Sample model: FreePortComm_RXTX.mdl (target)

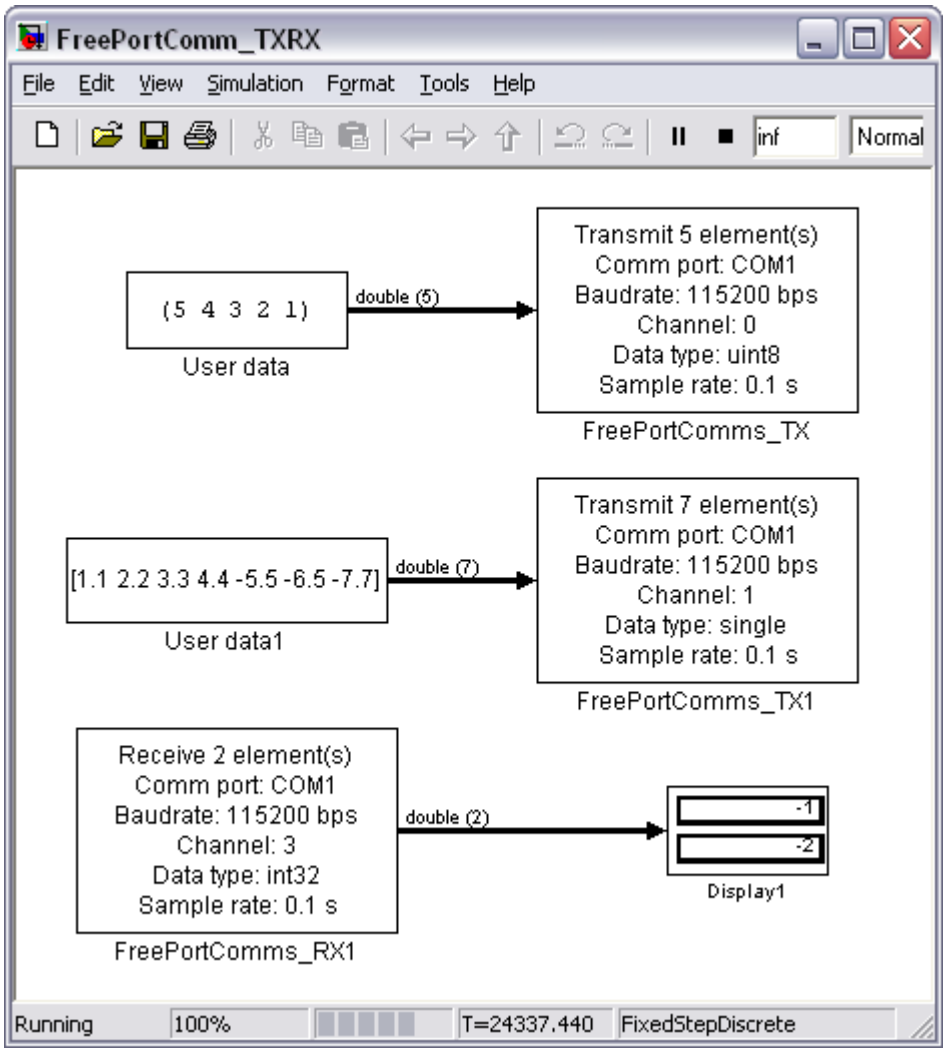

Figure 4-16 Sample model: FreePortComm_TXRX.mdl (host)

Note:

When using simultaneous upload and download it is important to *start the host model first*, then the target model. This is because the S-Function underlying the host sided *FreePortComms_RX* block flushes the reception buffer when it is started. If the target model is already running by the time the host model is started, it can happen that this flushing of the buffer on the host leads to the discarding of valid data bytes. As the FreePort system does not (yet) implement a proper host-target synchronization mechanism, it can happen that the data packet contents are misinterpreted. In this case, the *FreePortComms_RX* block is likely to emit garbage. Note that this particularity only happens with host models which include both FreePort receive as well as FreePort send blocks.

4.7.4 Download of unformatted data to the target

The target model *FreePortComm_RX_simple3.mdl* and the corresponding host model *FreePortComm_TX_simple3.mdl* demonstrate the download of raw data from the host to the target using the FreePort communication blocks. Figure 4-17 is the target-sided model, whereas Figure 4-18 shows the host-sided equivalent.

A similar pair of block diagrams can be designed for the upload of raw data from the target (microcontroller) to the host.

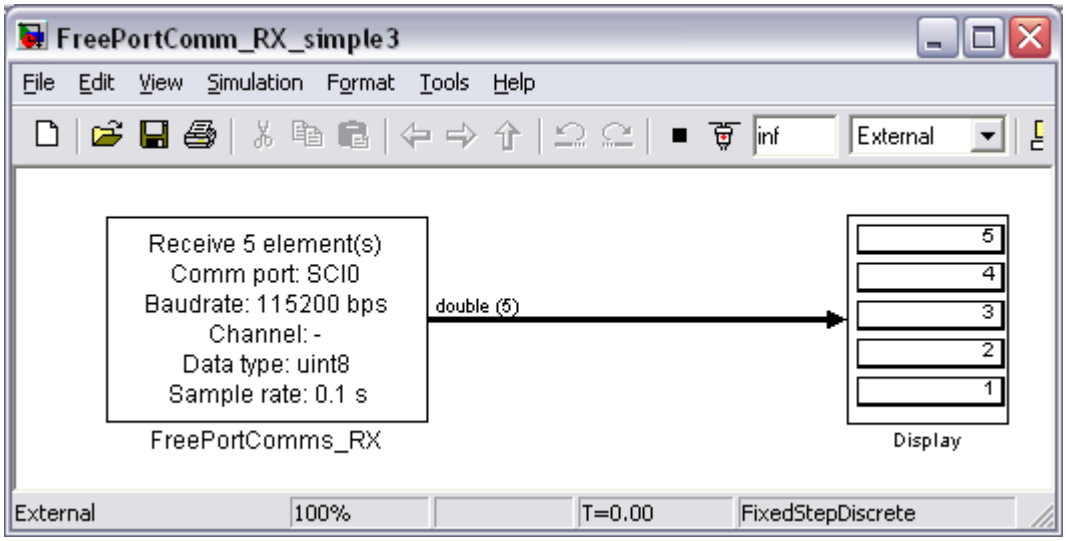

Figure 4-17 Sample model: FreePortComm_RX_simple3.mdl (target)

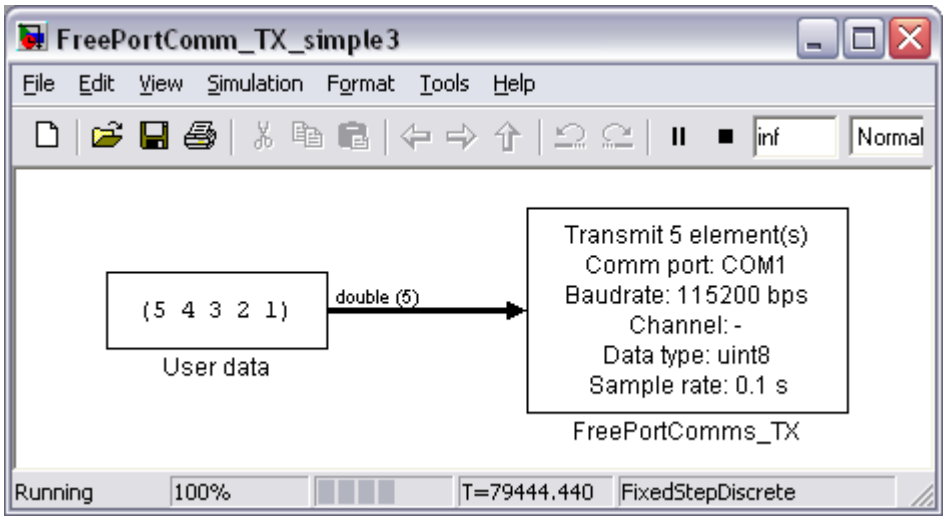

Figure 4-18 Sample model: FreePortComm_TX_simple3.mdl (host)

4.7.5 Upload and download of data via both ports SCI0 and SCI1

The target model *FreePortComm_RXTX_noExt.mdl* and the corresponding host model *FreePortComm_TXRX_noExt.mdl* demonstrate the download of data from the host to the target using the link from COM2 to SCI1 while simultaneously using the link from SCI0 to COM1 for data upload. Figure 4-19 is the target-sided model, whereas Figure 4-20 shows the host-sided equivalent. Notice that this requires the *External Mode* interface to be disabled (no background monitoring).

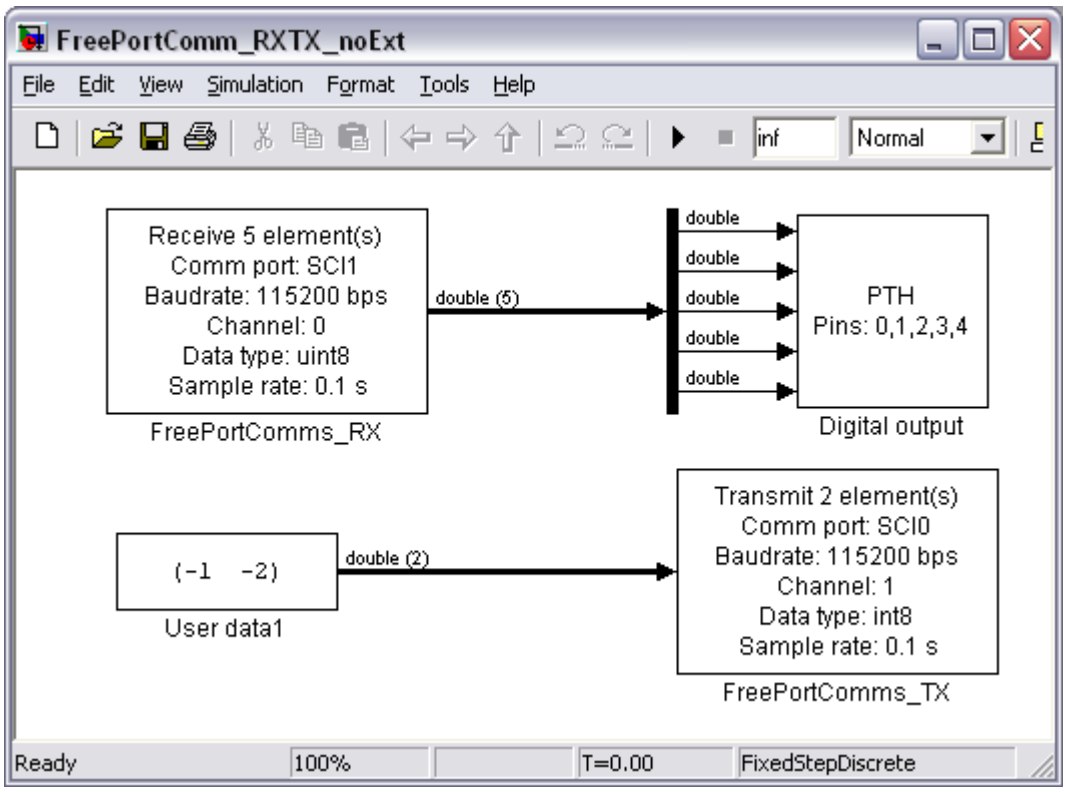

Figure 4-19 Sample model: FreePortComm_RXTX_noExt.mdl (target)

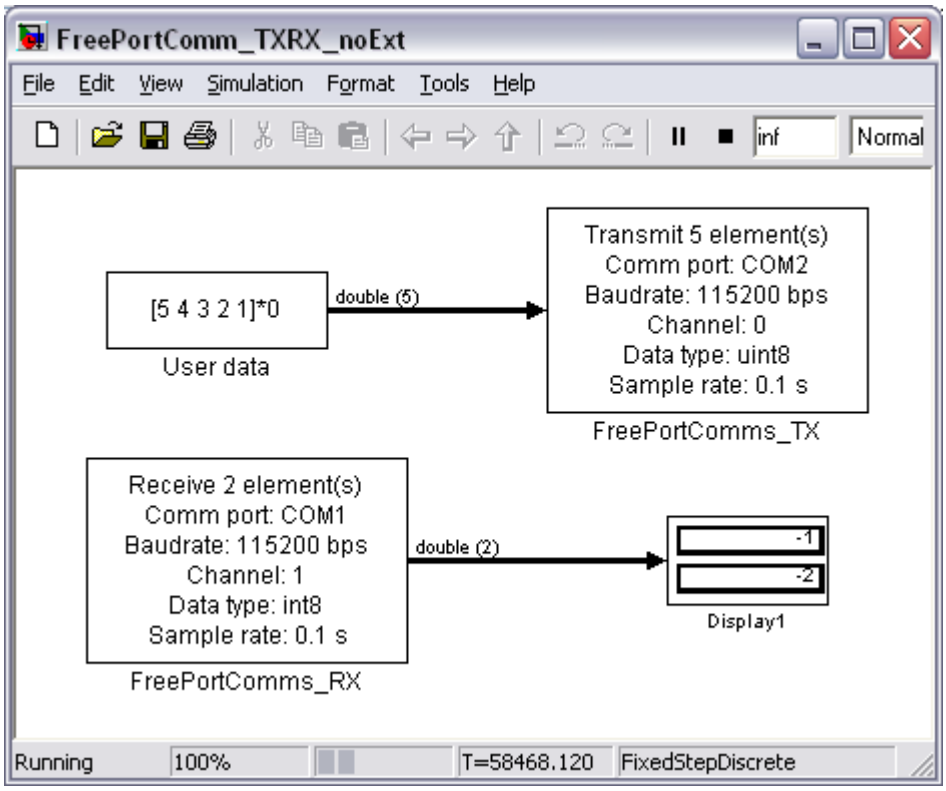

Figure 4-20 Sample model: FreePortComm_TXRX_noExt.mdl (host)

Remarks:

(1) When using simultaneous upload and download it is important to *start the host model first*, then the target model. This is because the S-Function underlying the host sided *FreePortComms_RX* block flushes the reception buffer when it is started. If the target model is already running by the time the host model is started, it can happen that this flushing of the buffer on the host leads to the discarding of valid data bytes. As the FreePort system does not (yet) implement a proper host-target synchronization mechanism, it can happen that the data packet contents are misinterpreted. In this case, the *FreePortComms_RX* block is likely to emit garbage. Note that this particularity only happens with host models which include both FreePort receive as well as FreePort send blocks.

(2) Using FreePort in parallel to the External Mode interface seems to require the target to be run out of ROM (target reset required – strange, but seems to be the case) to prevent the on-chip serial monitor from interfering with the FreePort interface SCI0.

(3) The FreePort communication blocks have been designed to cover a maximum number of situations. They can be used for serial communication between 2 hosts (e.g. COM1 on host A to COM1 on host B), COM1 to COM2 on one and the same host (loop-back operation), between a host and a target or between two different targets (e.g. microcontroller A to microcontroller B).