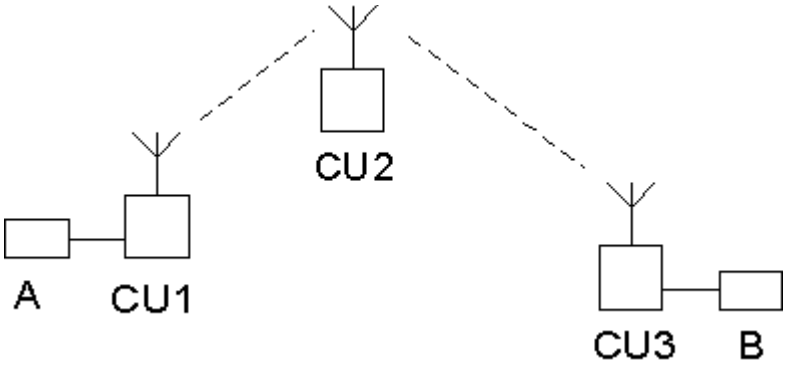
MORSE Guide

Table of Contents

What does a simple MORSE network look like.

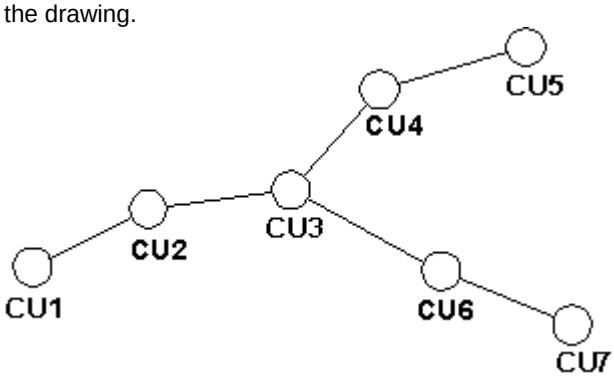
The following picture is used to give an initial idea about the MORSE network. Information transmitted in the form of data packets comes from the connected device A along the wire link to "radio" CU1. This "radio" is a complicated piece of equipment that checks the packet, modifies it and supplements it with records necessary for further transmission over the radio channel. This is why it is labelled as a communication unit (CU).



Simple network, CU detail

Amongst information added to the packet is the first section of the route which the packet will travel through via the radio environment to the destination CU3. In our example CU3 is out of reach of direct audibility of CU1, therefore the packet is routed through CU2. This communication unit, as for all the others, is equipped with intelligence, which is used to take care of the packet along the next section of its route. CU2 thus receives the packet from CU1 and sends it on to CU3. This method of transmitting packets can be called store and forward relaying. At the destination CU3 the packet is sent along the wire link to the external device B.

A radio network is generally more branched. Connection is made between individual points in the direction of lines marked in the drawing.



Branched network, CU detail

Besides these routes it is often possible to achieve connection by further combinations, for example between CU4 and CU6. Whichever links will be used during operation is determined by the quality of the connection, according to loading of the network and according to the structure of addresses used in the network. Each CU has information about the choice of routes in its routing table (also store and forward table) and according to these it then decides about routing obtained packets.

Connection-oriented and Connection-less Transmission of Data.

The network can transmit data in two ways. For connection-oriented transmission a route is created in advance by connecting individual sections of the network and the route created is then available for bi-directional transmission of data. An example is the telephone network. This method is suitable for the transmission of a large amount of data in one transmission. Other participants must wait during this period.

The second method is connection-less transmission which is also used in the MORSE network. Packets with data are sent to the network here. Each packet is provided with the destination address and travels through the network by itself. The length of the packet is limited to 1500 bytes, so transmission between two CU's runs from 0.3 to 1 second depending on the length of the packet and other conditions. Another packet can then pass through the same section of the route to another destination. The suitability of non-linked transmission for frequent routing of short messages to a large number of participants comes from this.

Non-linked transmission including addressing can be likened to the postal service. In this case information is sent with the address of the network of post offices which gradually regulate its progression to the addressee. Let's try and compare analogical terms:

Letter	Packet
Postal address	Packet header with address
Letter text	Data
Signature	CRC
Registered letter	Acknowledged receipt of packet
Letters can overtake each other	The same order of registered packets is not guaranteed
Sorting of letters before delivery	Routing of packets

Address

The example of the postal address can help us further - the address defines the addressee by the order of data:

```
country - town - street - name
```

The MORSE address also contains 4 levels for determining the destination in its 4 bytes. They are termed:

```
global - net - wide - local
```

An example of an address is given below:

```
69A1B2C3
```

It contains 4 bytes with each created from two hexadecimal symbols:

- 69 - global (country specification, 69=CR, 6A=Slovakia...)
- A1 - net (area in the country or other closer definition of the network)
- B2 - wide (subnetwork, further specification) su
- C3 - local (specific CU user)

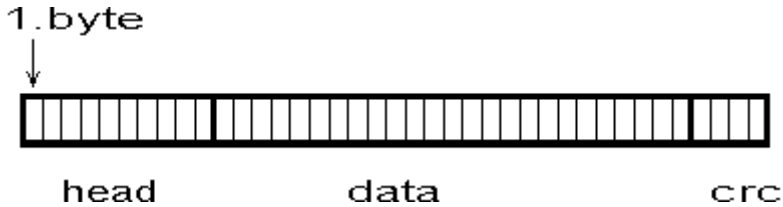
For example the subnetwork 69A1B200 can contain 256 local addresses 00 to FF, thus 69A1B200 to 69A1B2FF.
Each byte can acquire a value of 0 to 0xFF (0 to 255 dec), the overall theoretical number of addresses is thus 256^4 which is more than 4 billion. If not stated otherwise the MORSE address is recorded using eight hexadecimal symbols.
Work with the address is described in paragraph routing.

The internal layout of the modem from the point of view of configuration is contained in paragraph Communication unit MR25ET.

Explanation of terms used

What is a packet

A packet is a sequence of bytes ordered according to precise rules.



At the front of this sequence is a group of bytes which contains basic information about the packet (type, length, who determined for, etc) and this is called the head. Behind this other bytes follow which carry necessary information (data monitored by measuring devices, GPS position of vehicle, text message, command for remotely operated pump etc.). This part is labelled as data. A packet is generally ended with a CRC which serves for data integrity checking.

What is the range of the MR25

The range or audibility of the radio signal in the 400 MHz band is influenced by the shape of the terrain and other obstacles. Under unfavourable conditions (city, valley) the range of the MR25 is several hundred metres or several kilometres. Under favourable conditions (direct visibility, good antenna) good quality operation is possible to distances over 100 km.

Radio Network Medium

The medium used for packet transmitting enables such broadcasting of messages where the transmitted message can be picked up by more receivers. Therefore the message must carry information about which of the receivers must process it further. In our case it is the medium of radio signals or the Ethernet network medium. Other media having this attribute are, for example, air for dissemination of audio signals or the postal network for sending letters or parcels.
The other media type only allow signals to be transmitted between two participants and can be called a link media. An example is the wire communication line protocol RS232 which is used for the connection between the CU and external devices. Another example of a link medium for audio signals is a taught wire between two boxes which together create a child's telephone.

Dispersed Intelligence

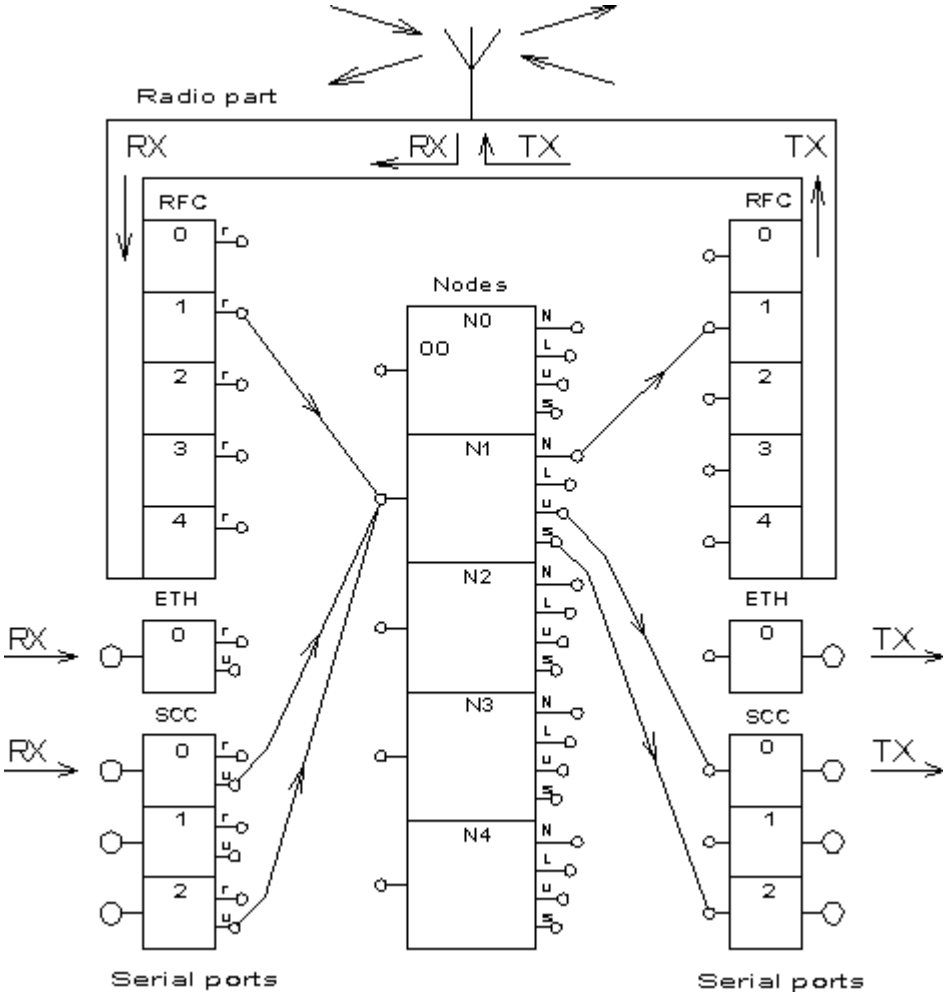
Operation in the network medium can be controlled in principle in two ways. The network can only contain one centre which controls routing of messages, for example a telephone exchange. Participants then only request the centre to mediate the routing of messages and the centre undertakes everything necessary.
The second possibility is that the network is composed of elements from which each is able to process the message in the necessary way. This means deciding where the message should be routed and route it to the appropriate channel (which here is the radio channel, Ethernet, wire communication line, output port, service channel) and check the data integrity. The MORSE system is composed of a CU equipped in this way. All CU are the same and differ only in the setting of configuration parameters. Therefore the MORSE network is easily expandable or adjustable.

How to proceed

Work with the address is described in paragraph routing .
The internal layout of the modem from the point of view of configuration is contained in paragraph Communication unit MR25ET .

Communication Unit MR25ET

The MORSE Communication Unit (CU or less precisely modem) can be represented from the point of view of configuration as:



CU layout, node detail

Individual blocks are always fitted with one input (from the left) and one or more outputs (to the right). Packets pass through the diagram from left to right. An exception is the antenna terminal above, which the received signal passes through into the CU and alternately the sent signal passes out through to the antenna.

In the centre there are **5 nodes**, which are the basic centre controlling communication. In reality they aren't boxes with terminal but parts of the program in the processor. From the point of view of configuration it is 5 independent control units, each with 1 input point and 4 outputs. Inside they are equipped with decision-making logic for controlling the flow of packets (routing). Each node can have a different address assigned to it under which it enters the MORSE network (thus one CU can be part of several independent networks).

The Radio part is marked in the upper part of the picture. It is common for the whole CU and individual nodes use it alternately. Its outputs pass through the radio-frequency channel (RFC), which according to the set rules enable routing of packets from nodes to transmitting radio parts. The receiving part of the RF channel transfers packets received from the RF medium to the node to which it is connected by the set configuration.

Serial Communication Channels (SCC) drawn in the lower part receive packets from external devices, modify them to the standard MORSE packet form and route them to the node. In its output part it then enables the output of packets from the node to the user device.

Ethernet Channel (ETH) enables input and output of packets between the Ethernet network and the MORSE network node.

The mentioned blocks are connected via communication lines according to the selection in configuration. Here, for example, at the input of node 1, 3 lines converge from RFC1, SCC0 and SCC2, and from its outputs 3 lines lead out in various directions. As a rule more lines can converge at the input however at the output only one direction of further progression of the packet can be defined. Lines are configured independently for receiving and transmitting. For example the connection from SCC0 to node 1 is different than from node 1 to SCC0

Notes on configuration

Nodes are numbered 0 to 5. Node 0 has an address firmly assigned. It is the serial number, which always starts with 00, for example 00572794. According to this number (address) it is possible to identify a CU at any time and by means of a so-called path-packets it can be used for establishing a connection. The address can be arbitrary assigned to other nodes and used for normal communication. There are four node outputs:

N	net	Network output to the radio network medium or Ethernet
L	line	Line output to wire link between two CU´s
u	user	User output for user data
s	service	Service output for Setr service terminal

SCC and ETH channels have a routing output, which is used for transferring packets inside the MORSE network and a user output used for packets which have come from outside to the MORSE network.
RFC channels have only a routing output for transmitting packets through the radio network. The user output would be of no use to the RFC.
Configuration parameters, i.e. the above-mentioned connections between channels and nodes and many other parameters that influence functions of the MORSE network are set using the service terminal. service terminal. It is a normal PC running the Setr.exe program, freely supplied by Racom.

Service terminal

The Service terminal is a program used for handling the CU of the MORSE system by a personal computer. It allows the setting of CU parameters, the monitoring of traffic on individual channels, the running of tests, the reading of records about past traffic and other functions. It is part of the applications group for handling a MORSE network and is accessible on www.racom.cz.

After choosing the SETR(DOS) DOWNLOAD on the main page you get the file `setr.zip`, which you can unzip best into the directory `c:\com\morse` on your PC. Next you can by choosing Firmware DOWNLOAD get the files `ma10.zip`, `mb10.zip` and `me10.zip` containing the software for the communication unit and unzip them into the file `c:\com\rfr`.

The directory `c:\com\morse` will be used for handling the CU. It contains among others the files `setr.exe, setr.par, morse.par, ver`.
The file **`ver`** contains a number that signifies the setr version, e.g. `554`.
The file **`morse.par`** contains the parameters which are common to others application belonging to the MORSE group. The parameter

```
#Comm. port
-p2
```

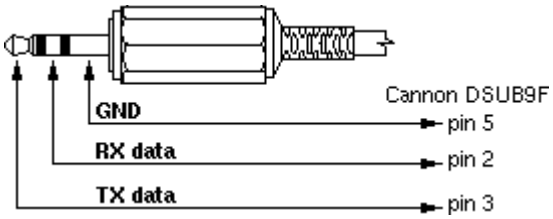
must be set to the number of serial port on the PC which will be used for the communication with the CU. Next it may be suitable to set the parameter

```
#set up the timezone
-t-3600
```

to the time difference in comparison with GMT. E.g. the middle Europe without the Summer time has the difference 1 hour, so we set the value -3600.
The file **`setr.par`** can contain only the parameters valid for the setr.exe. By default the parameters contained in the `morse.par` are valid, so the `setr.par` contains only the reference:

```
#Common parameters are in file morse.par.
+morse.par
```

The file **`setr.exe`** contains the software for handling the CU. It is necessary to connect PC to the CU with the service cable DKR12.



The wiring of the service cable DKR12.

We can start Setr.exe now. The display of service terminal appears. It contains on the bottom border from right:

- number of Setr version
- MORSE address of CU connected
- simple Help started by Alt+Z

The MORSE main menu is the main content of the display and the whole CU handling is started here.

MORSE main menu

contains all functions for controlling the CU.

```
MORSE main menu:

(H)W  (U)nit  (R)adio
(N)odes  (B)c  r(T)ab
(S)CC   R(F)C   (E)th
(A)rt

(s)ervice      d(i)ag
(p)ath        (?)help

(o)ld cnf menu

(q)uit
>>
```

Work with the menu and its options is described in the documentation MORSE Firmware. The first part of the menu serves for setting configuration parameters, the second for service and diagnostic work and the last allows older versions of software to be controlled.
Individual commands are selected by pressing the letter given in brackets. For example, entry of the node address is done as follows:

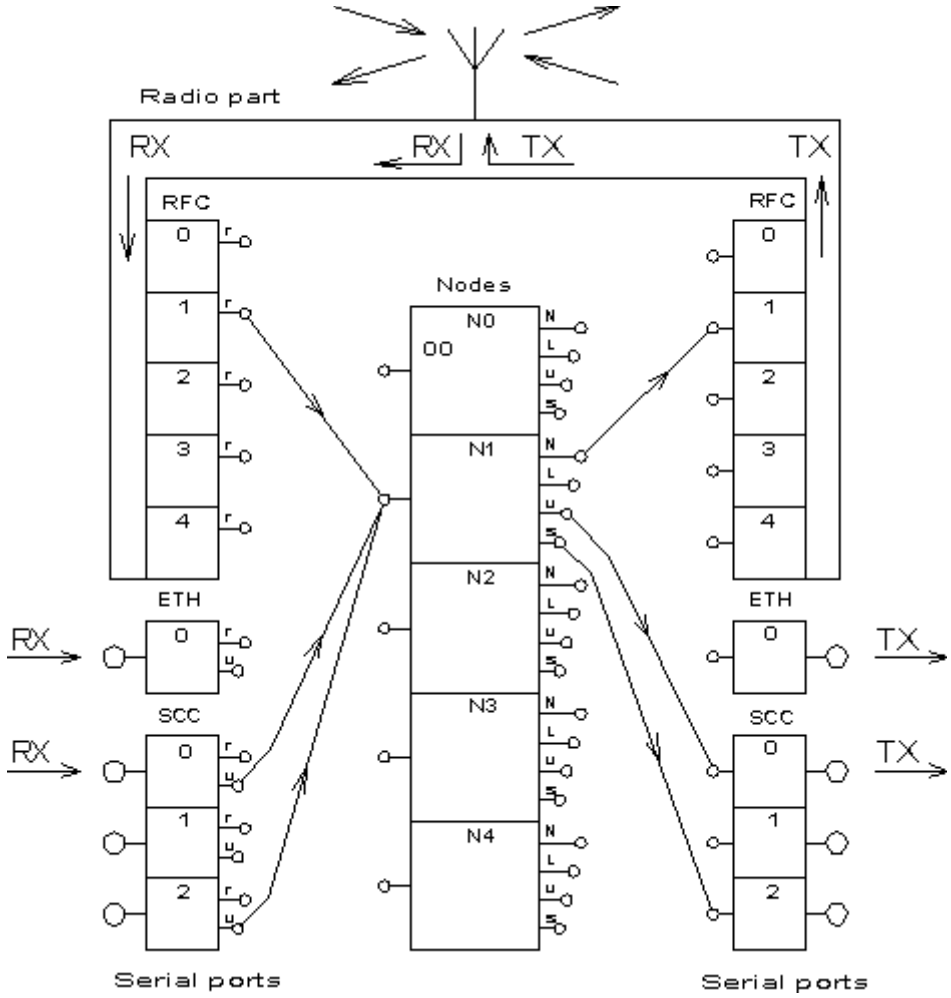
press **N** for selection of command `Node`, press `Enter`
press **e** for selection of command `edit`, press `Enter`
message ending with `O.K.` provides notification that the existing parameters were transferred from the CU to Setr.
press `Enter` to display them
now we can see the state of parameters of menu `Nodes`:

```
  Nodes:
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 00572794    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1109    S00 S02| -  R01|1 1 1 1| 15 SERV  OFF 304  30
(2) 00000000    S01 S02| -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

We can see that node 1 already has the address `690F1109` inserted and routing tables local, wide, net and global assigned, each with serial number 1. Other parameters have a recommended initial value, which is the value that appeared after selecting the command `de(f)ault`.
In order to insert for example address `69112233` to node 4,
press **4** `Enter`, then follow help on screen
press **a** `Enter`,
insert address **69112233** `Enter Enter`,
return back 1 level **q** `Enter Enter`,
now the new address which is currently only in Setr can be transferred to flash memory in the CU **w** `Enter`,
after receiving confirmation of entry `O.K.` we can write the new address using command **I** Enter also into RAM memory, thus setting it into operation.

The whole operation does not seem very transparent; however after gaining a little experience we can do this from `main menu` as follows:
Ne `Enter (O.K.) Enter`
4a 69112233 `Enter Enter`
w `Enter (O.K.)`
I `Enter (O.K.)`

Basic principles of configuration are quite evident from the CU layout diagram:



Connection from the node to the communication channel is set in the previously mentioned menu **Ne**, i.e. `(N)odes (e)dit`:

```
  Nodes:
                                  retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 0058D9E1    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 6950108F    S00 S02| -  R01|1 1 1 1| 15 SERV  ON  304  30
(2) 690F1209    S01 S02| -  R02|2 2 2 2| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30
```

Each node has one row and its columns **u,s,L,N** define where the respective output is directed. Row `(1)` corresponds to connections in the figure. Another 4 columns **l,w,n,g** determine using numbers 1 to 4 which of the prepared routing tables will be used by the node when resolving routing.

Connection from the SC channel to the node is selected in menu **SIe**:

```
Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1 MASK 00000000/08 ON  OFF NONE
(1) 0   NO AR         | 2 MASK 00000000/08 ON  OFF NONE
(2) 0   NO AR         | 1   NO AR          ON  OFF NONE
```

Again each channel has 1 row; the `retranslation` group of columns serves for routing output **r** and the `user` columns are determined for routing output **u**. Basic information is carried in column **N**, which determines which node the output is routed to. Rows `(0)` and `(2)` correspond to connections in the figure. Other items in the table will be explained in the SC channel properties definition.

Similarly menu **FIe** is used for routing outputs from the radiofrequency channel:

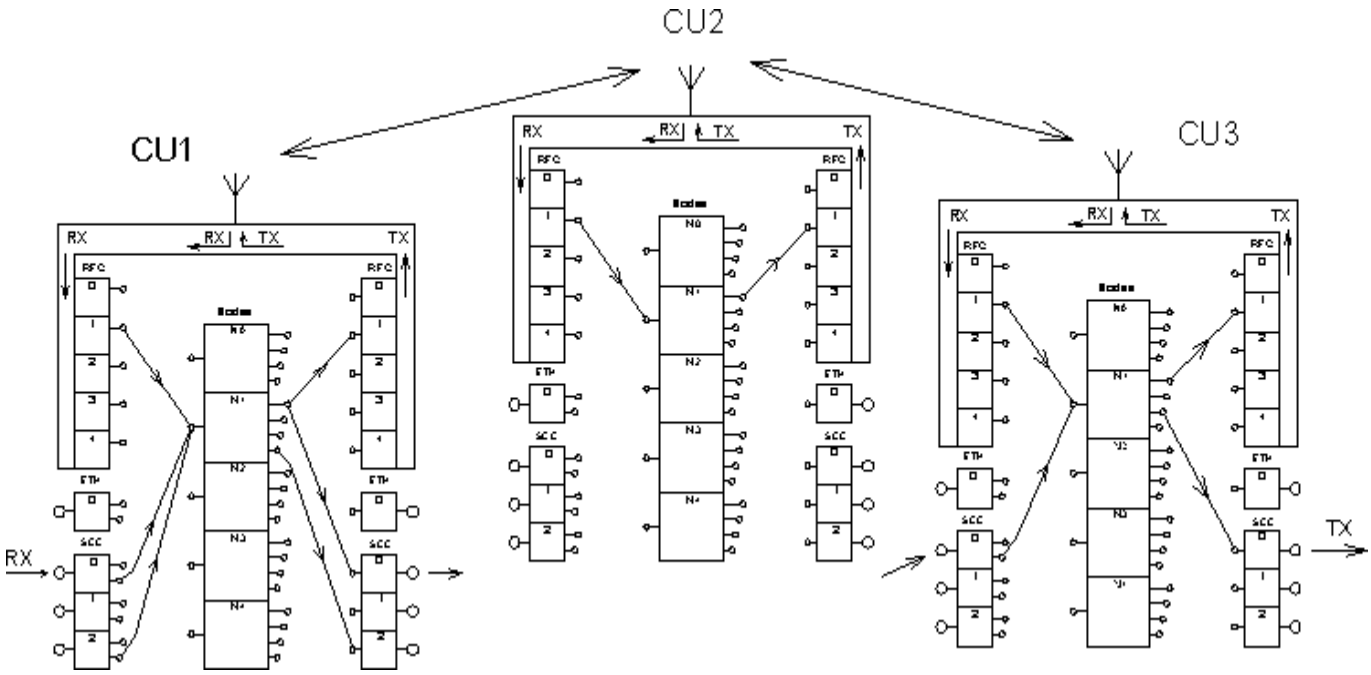
```
Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 0   NO AR          ON  OFF NONE
(1) 1   NO AR         | 0   NO AR          ON  OFF NONE
```

```
(2) 2    NO AR          | 0    NO AR          ON   OFF NONE
(3) 3    NO AR          | 0    NO AR          ON   OFF NONE
(4) 4    NO AR          | 0    NO AR          ON   OFF NONE
```

The only difference is in the number of rows, according to the number of RF channels, and in that the `user` columns are not used here. Row `(1)` corresponds to connections in the figure. In defining routes from the RF or SC channel to the node the address of this node is simultaneously assigned to the channel. If a packet is then received through a radio part then according to its address it is determined whether is pertains to any of the RF channels. If not then it is not processed any further. If yes then it is transferred to the node along the route configured in this menu.

Menu **`EIe`** behaves in the same way for the Ethernet channel.

For a better understanding of the above-mentioned configuration connections see the figure below of three communication units and connections, forming a simple radio network. Connections used along the packets route are also included:



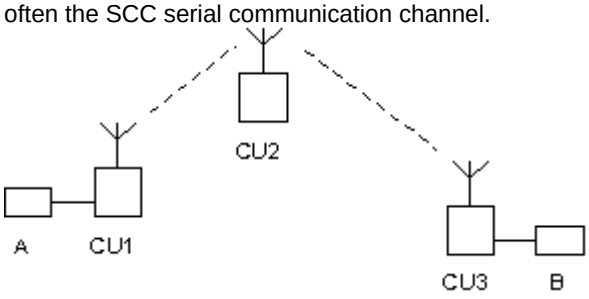
- The packet passes through channel SCC0 (received =RX) to CU1. Node N1 sends it via radio channel RFC1.
- Radio part CU2 receives the packet, channel RFC1 sends it to node N1 and from there it continues through the transmitting part RFC1 to the antenna. This step is termed store and forward relaying.
- At the destination CU3 the packet is received by the radio part, it is transferred to node N1 by channel RFC1 and from there to the serial channel SCC0, through which the packet leaves the MORSE network (sent through channel SCC0 =TX).

- The packets route in the opposite direction from CU3 to CU1 is analogical. It starts with the arrow entering SCC0 in CU3 and ends with the arrow leaving SCC0 in CU1.
- Connections connecting SCC2 with node N1 are also drawn. These serve for routing service packets (for example upon configuration setting) between Setr and node 1, because the service cable is connected to SCC2 for this purpose. Similar connections are in each CU, so that access from Setr to each communication unit is possible using the service cable.

- As shown in this example the node is the centre of the design-making operations for controlling the route of packets through the network. Introduction to routing gives a definition of these operations.

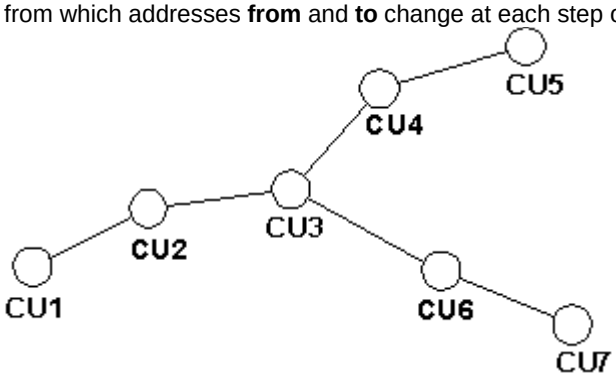
Introduction to Routing

A packet entering the MORSE network must be provided with a destination address. This address is assigned to it in the user device A or B. This external device can determine the address for example as only a serial number from 1 to 5 if it works in a network containing 5 stations. Transfer of addresses to the MORSE form is then taken care of in the input part of the CU, most often the SCC serial communication channel.



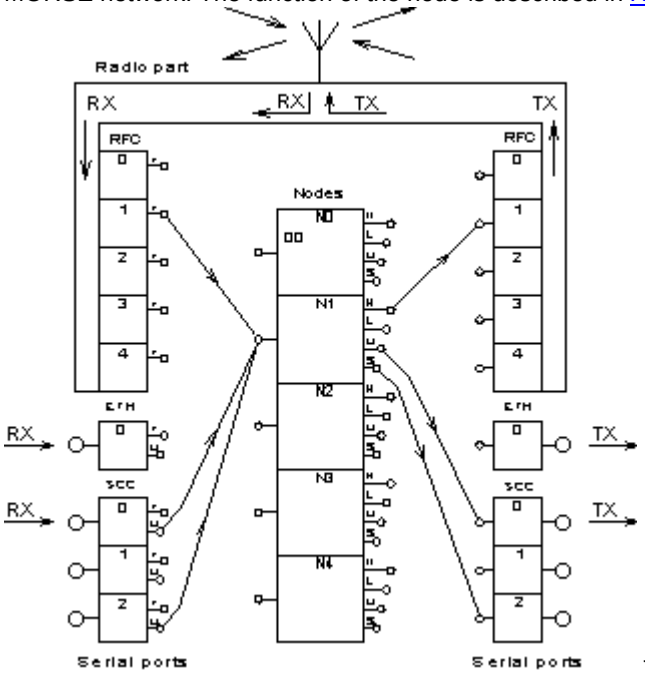
A simple network, detail CU is shown below.

This destination address (shortened to dest or **dst**) is carried in the header of the packet along the whole route through the network. It serves as initial information for each CU during routing, i.e. upon determining the next step of the packets route. The packet also carries a source address with it (**src**), according to which the destination CU recognises whom it should send a reply. For example, see next drawing, a packet sent in the subnetwork 69AABB00 from CU1 to CU6 thus has the **src** address 69AABB01 and **dst** address 69AABB06. The route of the packet is divided into a number of steps between neighbouring CU, here 1 > 2, 2 > 3, 3 > 6. For each of these steps routing is resolved separately and the packet takes these two addresses of the current step with it - **from** is the transmitting CU and **to** is the receiving CU. Addresses contained in the header of the MORSE packet are thus:
src, from, to, dst
from which addresses **from** and **to** change at each step on route.



The branched network, detail CU is shown below.

All routing operation run in any of the 5 nodes contained in the CU. In the node the address **to** is assigned to the passing packet in the manner described below and the packet is sent via any of the four node outputs on its next route through the MORSE network. The function of the node is described in Routing 1.



The communication unit CU, node detail is shown bellow.

For illustration here we give a record of the route of the signal through three nodes from address `690F120F` to address `690F1209` and back:

```
u S02   690F120F    R01
30/ 46  690F1200    R01
31/ 88  690F1209   serd

 serd   690F1209    R02
30/ 41  690F1200    R01
30/ 65  690F120F  u S02
690F1209h>
```

in the middle the route is marked by the sequence of addresses, on the right the channel is marked through which the packet left the node and on the left there is the strength of the received signal.

MORSE addresses thus have various functions according to their place on the route of the packet, as has just been explained.

From the point of view of the internal construction an address is composed of 4 bytes called global, net, wide and local, see introductory chapter. According to these parts of the address the node searches for the further information in the routing (or store and forward) tables. The order of these node activities is discussed in the following chapter Routing 1. Here you can see work with the routing tables.

From the MORSE main menu selecting:
`>> T Enter` we get:

```
Retranslation table:
  (l)ocal
  (w)ide area
  (n)et
  (g)lobal
(q)uit
>>
```

We select which type of table we would like to work with, for example for table wide:
`>> w Enter`

```
Wide retranslation table No:
  (1) (2) (3) (4)
(q)uit
>>
```

The table can be prepared in up to 4 versions so that each node can have a different table available. Select table number one:

```
>> 1 Enter

Retranslation table:
  (r)ead nontrivial paths
  (p)ath:0 via (n)ode:0
  (g)et p(u)t
  (c)lear  (e)dit
(q)uit
>>
```

Using the function `(r)ead` we can read the contents of the table:
`>> r Enter`

```
Wide retab. No 1
14to:1105 15to:150A
>>
```

Using the function `(c)lear` we can delete the whole table. An entry is made using `(p)ath` and `(n)ode` with a subsequent `(p)ut`. If an entry is made using the standard procedure as follows:
`>> p 14 Enter`
`>> n 1105 Enter`
then it is not necessary to use the command `p(u)t`.

Upon writing to individual tables this length of address is used for the item `(n)ode` :

```
global 69112233
net    69112233
wide       2233
local        33
```

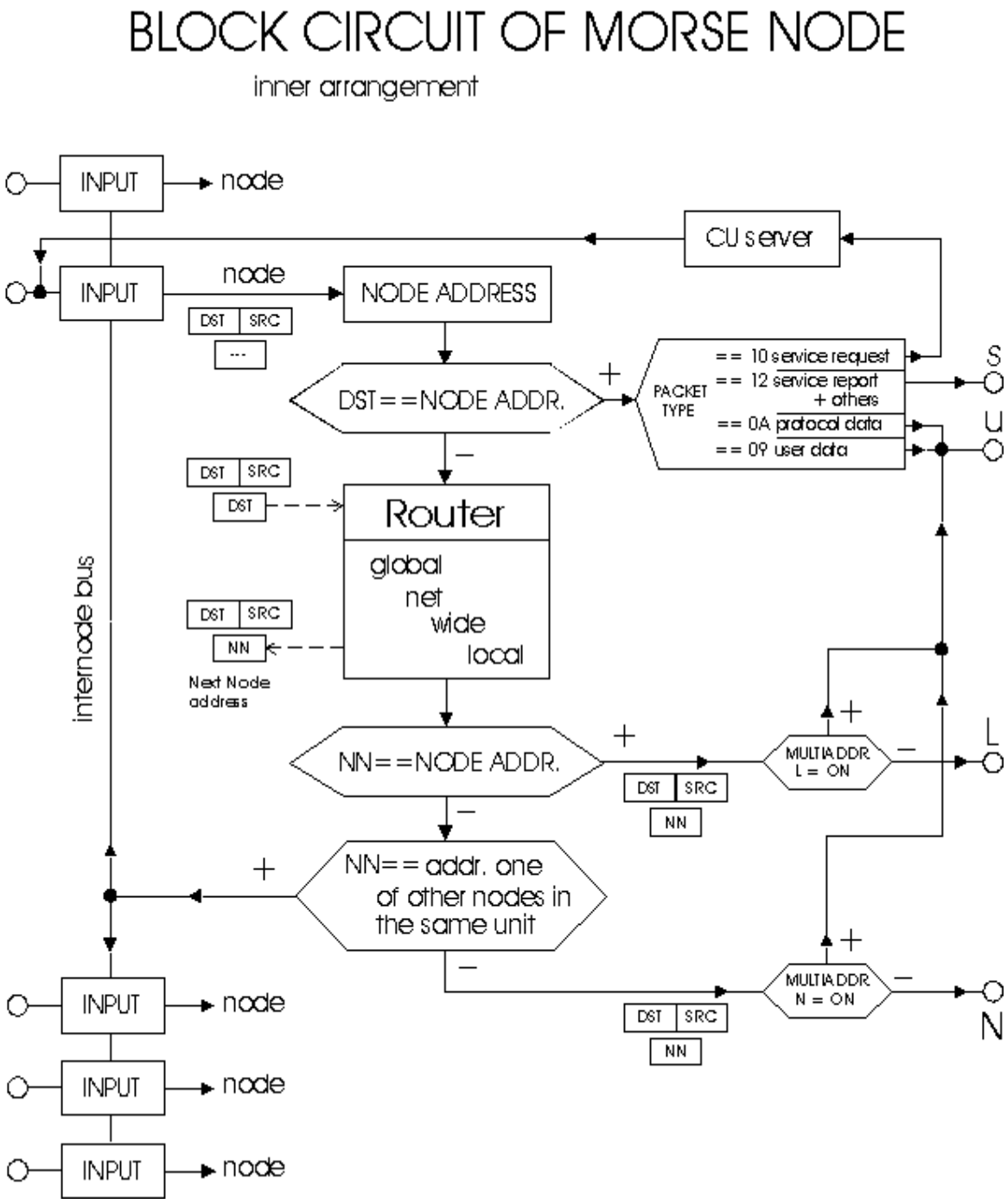
The item `(p)ath` always has the length of one byte, e.g. `22`.

Rules for creating the content of routing tables are described in the following chapter routing 1.

Packet through Morse node - simplified description.

Packet incoming to the node, contains it's target address DST. The address of immediate next step NN is determined in the node according to address DST.

The address DST is compared with address of the node (NODE ADDRESS), where the process runs. If they are equal then the packet is in the target node and will be sent, according to its type, to the user or service port or to the service processing. If they are different, then the routing tables processing follows.



The packet address can be divided to the 4 parts called (from left) global, net, wide and local. The "global" part represents a big unit where the address belongs (country), part "net" determines more exactly the region inside the country (geographically or according to other points of view). Next specifying is in part "wide" and part "local" defines finally the actual address of node. At the choice of routing table the addresses "dest" and "node addr" are compared. The parts global, net, wide and local are compared gradually. If the difference is found (for example in "net" part), then in the respective table (net) will be the address of next step found. The differences in lower parts (here wide and local) are then not important.

The routing table contains the pairs (p)ath and (n)ode. The (p)ath represents the compared part of the address (here net) and (n)ode is address of node NN where leads the next step of packet on it's way to DST address. The tables wide and local contains in (n)ode the lower address part only. The item (n)ode is the resultant information found in the routing tables and is

inserted into address NN.

The packet equipped now by the NN address goes to the next processing according to diagram "Block circuit of MORSE node". If the NN address equals to NODE ADDR, then the packet is sent to the link output L.

In next step is the address NN compared with addresses of other nodes in own CU. If the concordance is found, then the packet is sent through inner bus to the respective node.

If the packet is no diverted to the link output or inner bus, then it is sent to the net retranslation output N.

In this way the packet is sent to the net medium (most frequently radio channel) and for it's next processing takes charge this node, which address is equal to TO address contained in the packet. The address NN becomes then the next step adrress TO in the MORSE network.

The more detailed description is in chapter Routing 2.
The particular routing examples follows in the next chapter.

1.1. Direct connection between two CU.

Let suppose, that in the communication unit (modem, radio) there is set the working frequency, power and that all parameters have the default value. Then we set the address `690F1230` in the menu `Ne` in one of them:

```
  Nodes:
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004CA177    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1230    S00 S02| -  R01|0 0 0 0| 15 SERV  OFF 304  30
(2) 00000000    S01 S02| -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

Save all by the command `(w)rite` and `(I)nit`. After new address setting initialize the connection with Setr by command `Alt+I`. The same things make in the opposite station with tis different, that the putted address is `690F1233`. Then give back the service cable in the first CU, perform `Alt+I` and write:

```
    >>!h690F1233 Enter
```

Suffice it to write the last part of address only, which is different from the own or the last called address:

```
    >>!h33 Enter
```

Now put the command for the packet sending:

```
    >>! Enter
```

The packet is sent there and back and the connection report appears:

```
u S02   690F1230    R01
31/ 77  690F1233   serd

 serd   690F1233    R01
30/ 75  690F1230  u S02
690F1233h>
```

The address column describes the way of packet. On the left there is the incomming signal description (signal source or for radio signal it's quality and strength DQ/RSS). On the right there is the direction in which the packet leaved the node.

1.2. Store and forward connection.

In this case we need three CU. The assigning of routing tables put in the CU `690F1230` . It would be sufficient to put the local table only but it is suitable to write all of them:

```
  Nodes:
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004CA177    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1230    S00 S02| -  R01|1 1 1 1| 15 SERV  OFF 304  30
(2) 00000000    S01 S02| -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30
```

We prescribe it the local table in CU `690F1230` , that the packets designated for the address `690F1233` should be sent to address `690F1200` which is audible for both station:

```
Local retab. No 1
33 to:00
>>
```

After command `!h33 Enter` we obtain next report:

```
u S02   690F1230    R01
31/ 51  690F1200    R01
31/ 68  690F1233   serd

 serd   690F1233    R01
31/ 69  690F1230  u S02
690F1233h>
```

We can see, that the packet from `690F1200` to `690F1233` runs through the store and forward station `690F1200` but in the opposite direction remained the direct route, because the routing table is not fullfiled in `690F1233` and the stations can hear each other directly.

Now put also in CU `690F1233` the local routing table

```
Local retab. No 1
30 to:00
>>
```

and assign the set of tables `1 1 1 1` also to the address `690F1233` . The packet runs now in both directions through `690F1200:`

```
u S02   690F1230    R01
28/ 69  690F1200    R01
31/ 66  690F1233   serd

 serd   690F1233    R01
29/ 50  690F1200    R01
31/ 71  690F1230  u S02
690F1233h>
```

Now both end points can be out of touch (the radio signal between them is too weak) but if each of them can hear the station `690F1200`, they can communicate together.

1.3. Distant access.

The configuration modifications can be done by service cable but more often the distant access through the radio channel is used. We remain connected by the service cable to the local station and hand on the service packets by radio channell to the distant station. If we obtained in the last example after command `>>!h33 Enter` this response, then the last line

```
690F1233h>
```

announces, that we are switched in the `690F1233` station and all commnads, we are putted in, will be performed in this station. Notice the change shape of prompt from `>>`, which refers to local access to `690F1200h>` denoting the access in the distant station. To return to the local access type:

```
690F1233h>!l Enter
```

Notice the longer response to service command for reading and writing into memory in the distant CU. Switch to the distant station

```
>>!h33 Enter Enter
```

and choose from the main menu

```
690F1233h>Ne Enter
```

Now takes place the service packet exchange between the stations, which takes according to the connection conditions from several tenth of seconds to several seconds and then the report appears

```
get  N0 N1 N2 N3 N4  O.K.
690F1233h>
```

which says, that the menu Node edit was succesfully transfered from the distant station into our program Setr. které sděluje, že menu Node edit bylo úspěšně přeneseno ze vzdálené stanice do našeho programu Setr. Display this menu by `Enter` and write e.g. the new address and routing table assignement in the Node 2.

```
  Nodes:
                                  retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004C9D8F    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1233    S00 S02| -  R01|1 1 1 1| 15 SERV  OFF 304  30
(2) 690F1234    S01 S02| -  R02|1 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F1233h>
```

These changes are yet done in our Setr and only by command `(w)rite` we send them into distant station. We obtain the acknowledgement about succesful writing in the flash memory:

```
write N0 N1 N2 N3 N4  O.K.
690F1233h>
```

Now we perform similarly the writing in the RAM memory by command `(I)nit` and in this way we initialize the putted parameters in the distant station.

```
put   N0 N1 N2 N3 N4  O.K.
690F1233h>
```

It could happend at bed connection or heavy radio traffic, that the acknowledgement `get, write` or `put` doesn't come. We repeat the command in this case. We return by command `!l Enter` in the local station, refill the local routing table to the state

```
Local retab. No 1
33 to:00  34 to:00
>>
```

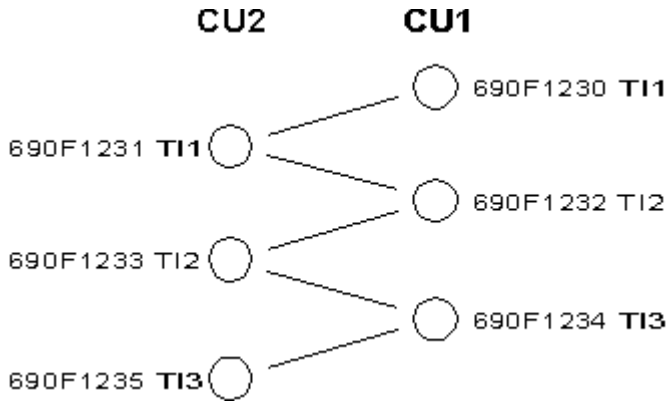
and by command `!h34 Enter` examine the new connection:

```
u S02   690F1230    R01
31/ 57  690F1200    R01
31/ 68  690F1234   serd

 serd   690F1234    R02
30/ 54  690F1200    R01
30/ 68  690F1230  u S02
690F1234h>
```

The connection through more stations is described in the chapter Routing examples 2.

2.1. Retranslation in more steps.

We will establish the simulation of a radio link composed of 5 conected radio steps between two CUs. Beside the addresses there are the number of local retranslation tables written, which are assigned to the nodes.



Each of nodes behaves like a separate CU. The configuration of the first CU:

```
  Nodes:
                                  retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004CA177    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1230    S00 S02| -  R01|1 0 0 0| 15 SERV  OFF 304  30
(2) 690F1232    S01 S02| -  R02|2 0 0 0| 15 SERV  OFF 304  30
(3) 690F1234    S02 S02| -  R03|3 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30
```

The content of the routing tables:

```
>>Tl1r

Local retab. No 1
35 to:31
>>Tl2r

Local retab. No 2
30 to:31  35 to:33
>>Tl3r

Local retab. No 3
30 to:33
>>
```

The configuration of the second CU and its routing tables:

```
  Nodes:
                                  retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004C9D8F    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1231    S00 S02| -  R01|1 0 0 0| 15 SERV  OFF 304  30
(2) 690F1233    S01 S02| -  R02|2 0 0 0| 15 SERV  OFF 304  30
(3) 690F1235    S02 S02| -  R03|3 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F1231h>Q

690F1231h>Tl1r

Local retab. No 1
35 to:32
690F1231h>Tl2r

Local retab. No 2
30 to:32  35 to:34
```

```
690F1231h>Tl3r

Local retab. No 3
30 to:34
690F1231h>
```

The connection from CU1 to CU2 through complete path:

```
>>!h35

690F1235h>!

u S02   690F1230    R01
30/ 70  690F1231    R01
31/ 70  690F1232    R02
31/ 70  690F1233    R02
31/ 70  690F1234    R03
31/ 69  690F1235   serd

 serd   690F1235    R03
30/ 70  690F1234    R03
30/ 69  690F1233    R02
30/ 70  690F1232    R02
31/ 69  690F1231    R01
31/ 70  690F1230  u S02
690F1235h>
```

The packet goes through the steps according to the routes in the routing tables to the destination node and returns to the source node as we can see in the middle column. To the right of the addresses, in the right column, are written the medium through which the packet left the respective nodes. These are the radio channels which are assigned to the nodes in the default configuration menu `(N)ode (e)dit` , the "service dispatcher" `serd` which is at the destination station and, on the end, the user channel `u S02` to which is connected Setr.
To the left of the addresses,in the left column, is always a characterization of the signal source from which the signal enters the node. Pertaining to our example, a packet may enter a node through an SC channel or an RF channel. For the SC channels it is a source `u S02` or `serd`, for the RF channels the source is not displayed rather it is the quality and the signal strength DQ/RSS. The DQ is a quality of received signal (a value of 31 is the best quality, 25 is a good usable quality, 15 is a less-quality signal, e.g. from reflection). RSS (Received Signal Strenght) is the signal strength (60 is a very strong signal, 90 is good and useable, 110 is a little signal on the boundary of usage). In our case it is all the time the same physical channel, so the signal strenght is the same always.

The routes in the routing tables of the nodes only routes packets which are going to the end points of the path, i.e. to the address `690F1235` or `690F1230`. If a packet is sent to any other point, it is transmitted to the `destination address` directly without the use of any routes. This would not be possible in a real network, since it will be composed of distant points. In our case, a connection between 690F1230 and 690F1233 is completed and looks like:

```
u S02   690F1230    R01
31/ 70  690F1233   serd

 serd   690F1233    R02
30/ 70  690F1232    R02
31/ 69  690F1231    R01
30/ 70  690F1230  u S02
690F1233h>
```

The next case is when the destination address is lying in the same CU. For example, the route to the address `34` is not written in the table of node `690F1230` and the packet is sent directly to the address `690F1234` . At the return path from `690F1234` to `690F1230` , the route to `30` is found in the table of node `690F1234` , so the packet goes through the prepared path.

```
u S02   690F1230     -
   -    690F1234   serd

 serd   690F1234    R03
30/ 70  690F1233    R02
30/ 70  690F1232    R02
30/ 69  690F1231    R01
30/ 70  690F1230  u S02
690F1234h>
```

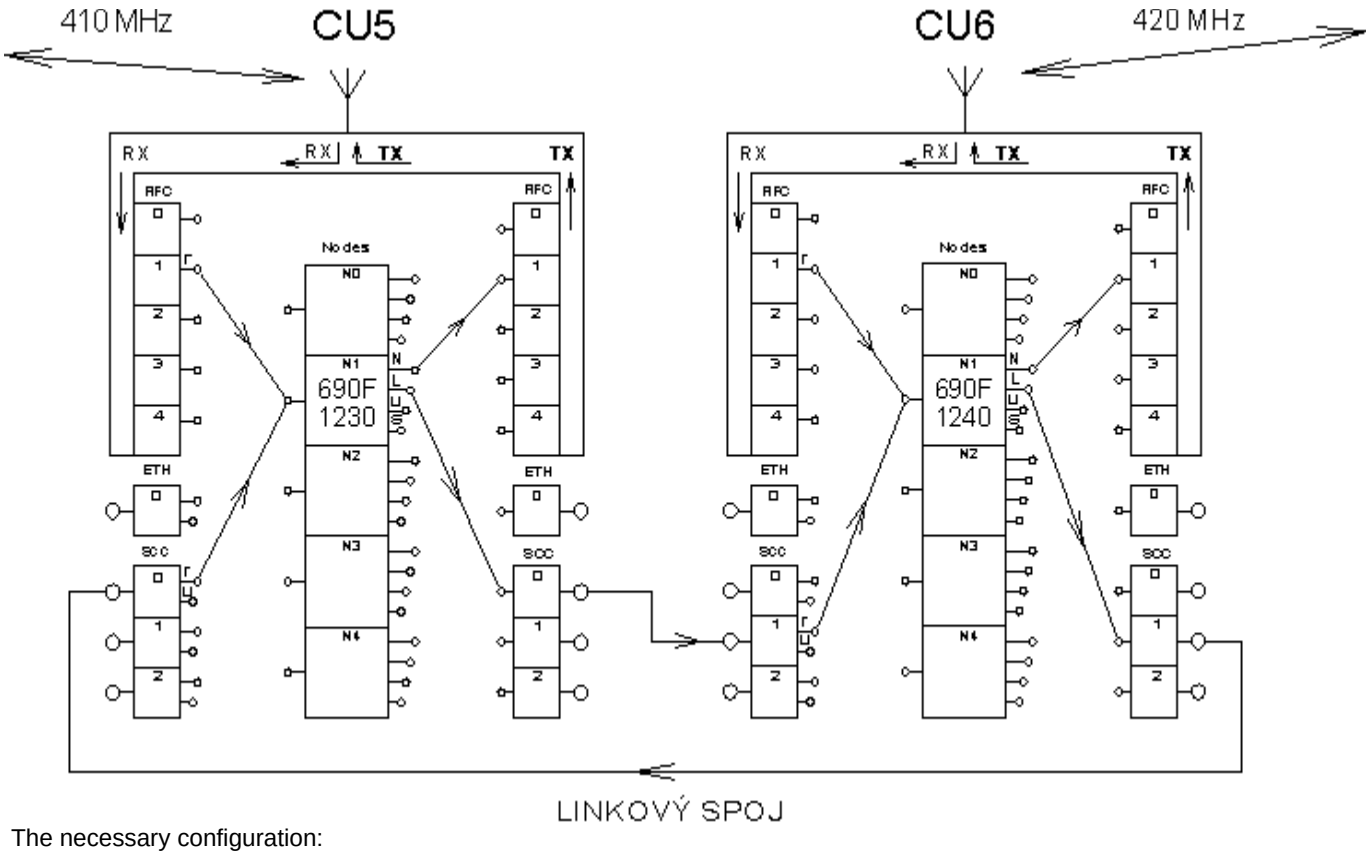
In a real large-space network, it should be possible for everyone to connect to everyone. Thus, the routing tables of every node must contain the instructions for routing to all other nodes. Only the trivial connections are omitted, this is the nearest node which can be called directly. The complete routing tables for our example follows. The first group belongs to the CU1 containing the address 690F1230, and the second to the CU2 containing 690F1231:

```
Local retab. No 1
32 to:31  33 to:31  34 to:31  35 to:31
>>Tl2r

Local retab. No 2
30 to:31  34 to:33  35 to:33
>>Tl3r

Local retab. No 3
30 to:33  31 to:33  32 to:33
>>!h31

690F1231h>Tl1r

Local retab. No 1
33 to:32  34 to:32  35 to:32
690F1231h>Tl2r

Local retab. No 2
30 to:32  31 to:32  35 to:34
690F1231h>Tl3r

Local retab. No 3
30 to:34  31 to:34  32 to:34  33 to:34
690F1231h>
```

In the next chapter there are the examples of connection on serial line. .

3.1. Serial line

The packet transmittion between two CUs by a serial cable (called the wire retranslation too) is used to connect two parts of a network working on different frequencies.
In our example, the packet comes through the network on the frequency 410 MHz into CU5, then it goes through RFC1 to the node N1 having the address 690F1230. From there by the link output L to SCC0 and then by the serial link to CU6. It inputs CU6 through SCC1, by the retranslation output r of SCC1 to the node N1 having the address 690F1240, by the net output N through RFC1 to the antena and then continues on the frequency 420 MHz.



The necessary configuration:

- The link output is routed to chosen SCC. The retranslation output from SCC is routed to the node.
- The protocol MARS-A is configured on the respective node.
- The way to the own address is written in the routing tables, see the description in the Routing 1.
- The serial channels of both CUs are connected by the three-wire link (RX,TX,GND). The RX clamp on one CU is connected to the TX clamp on the others, so called crossed line.

- The example of configuration CU5:

```
  Nodes:
                                    retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004C9D8F    S02 S02|  -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1230    S00 S02|S00 R01|1 1 1 1| 15 SERV  OFF 304  30
(2) 00000000    S00 S02|  -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S01 S02|  -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02|  -  R04|0 0 0 0| 15 SERV  OFF 304  30


Channel to Node Interface:
    retranslation     |   user                          lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 1   NO AR        | 1 MASK 00000000/08 ON  OFF NONE
(1) 0   NO AR        | 2 MASK 00000000/08 ON  OFF NONE
(2) 0   NO AR        | 1   NO AR          ON  OFF NONE


Local retab. No 1
40 to:30
```

- The example of configuration CU6:

```
  Nodes:
                                  retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004CA177    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F1240    S00 S02|S01 R01|1 1 1 1| 15 SERV  OFF 304  30
(2) 00000000    S01 S02| -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30


Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1 MASK 00000000/08 ON  OFF NONE
(1) 1   NO AR         | 2 MASK 00000000/08 ON  OFF NONE
(2) 0   NO AR         | 1   NO AR          ON  OFF NONE


Local retab. No 1
30 to:40
```

- The next response after sending of command ! is valid for the simple case, when the Setr is connected to CU5 and the CU6 is called:

```
690F1240h>!

u S02   690F1230    S00
  S01   690F1240   serd

 serd   690F1240    S01
  S00   690F1230  u S02
690F1240h>
```

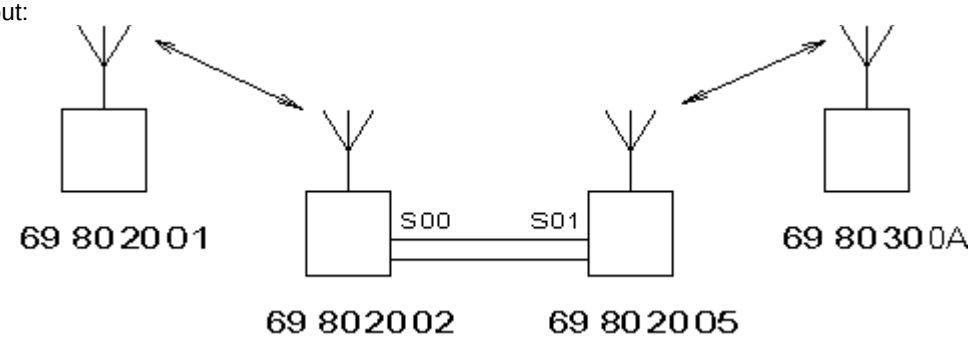
3.2 The serial line connected to the network

An example with the serial line and with the addresses containing two different wide items.

The record of packet passing is contained in its left column. The entry DQ/RSS appears when the packet arrived through radio channel or the marking of serial channel, when the packet arrived by serial channel. In the right column is the marking of channel through which the packet left the node.

```
u S02   69802001    R01
30/ 56  69802002    S00
  S01   69802005    R01
25/ 46  6980300A   serd

 serd   6980300A    R01
27/ 42  69802005    S01
  S00   69802002    R01
30/ 56  69802001  u S02
6980300Ah>
```

The network layout:



The comment to the content of the routing tables:

Tables in the node 69802001:

- it is not necessary to route packets to node 69802002 because their addresses are different in local part only and there is a direct radio connection.

- the way to the node 69802005 is routed to 69802002, the record is:

```
Local retab. No 1
05 to:02
```

- the route to node 6980300A, whose address is different in the wide part, will be looked for in the wide table:

```
Wide retab. No 1
30to:2002
```

Tables in the node 69802002:

- it is not necessary to route packets to node 69802001
- the routes from node 69802002, which is behind the serial link (so called "behind wires"), to the nodes 69802005 and 6980300A must be routed to its own address, even if the addresses are different in the local part only:

```
Local retab. No 1
05 to:02
```

- the route to node 6980300A will be looked for in the wide table, because the address 6980300A differs from the own one in the wide part:

```
Wide retab. No 1
30to:2002
```

Tables in the node 69802005:

- the routes to the nodes 69802001 and 69802002 are identical to that of the input to the wire link, so the routess in the local table are routed to own address:

```
Local retab. No 1
01 to:05  02 to:05
```

- the route to the node 6980300A is looked for in the wide table:

```
Wide retab. No 1
30to:300A
```

Tables in the node 6980300A:

- all nodes to which we will route the packets are lying in other wide than 30, so routes in wide table is required. Since all other nodes are lying in a common wide, 20 , only one route suffices:

```
Wide retab. No 1
20to:2005
```

After this practice of routing we give again the description of process how the packet is going through the node completed with more details.

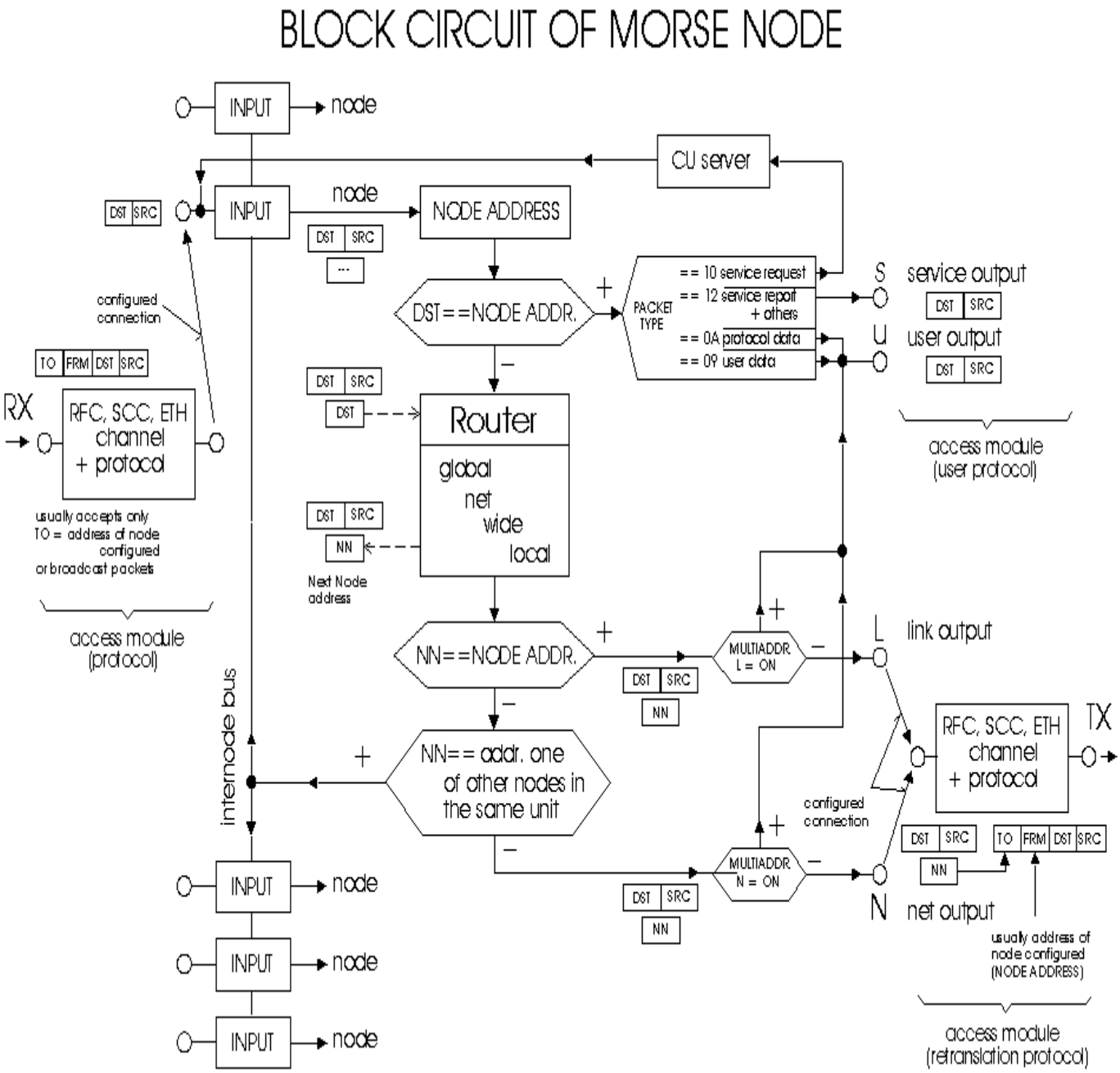
Packet through Morse node - enlarged description.

The routing packet entering the CU carries a quadruple MORSE address - source, SRC, destination, DST, node address from where it is coming, FRM, and a node address to where it is directed, TO. The procedure of processing is represented in the diagram "Block circuit of MORSE node" and includes the content of the mentioned addresses.

The access module, represented in the left part of the diagram is assigned by configuration to some of the nodes. The module only accepts packets whose TO address is the same as the address of that node. From there the packet continues with only the SRC and DST addresses.

Packet incoming to the node, contains it's target address DST. The address of immediate next step NN is determined in the node according to address DST.

The address DST is compared with address of the node (NODE ADDRESS), where the process runs. If they are equal then the packet is in the target node and will be sent, according to its type, to the user or service port (including the SRC and DST addresses) or to the service processing. If they are different, then the routing tables processing follows. Into address NN is at first inserted the address DST, which is later rewrited by address found in respective routing table.



You can get the diagram in pdf format.

The packet address can be divided to the 4 parts called (from left) global, net, wide and local. The "global" part represents a big unit where the address belongs (country), part "net"
determines more exactly the region inside the country (geographically or according to other points of view). Next specifying is in part "wide" and part "local" defines finally the actual
address of node. At the choice of routing table the addresses DST and NODE ADDR. are compared.
The parts global, net, wide and local are compared gradually. If the difference is found (for example in "net" part), then in the respective table (net) will be the address of next step found. The differences in lower parts (here wide and local) are then not important.

The routing table choosed in this way, can be in the CU (Communication Unit) prepared in the
versions 1 to 4. This version is used, which number is written in the menu (N)odes (e)dit in the row of respective node and in the column according to table type (here net). If in this place is not the number 1 to 4 written then on the place of address NN remains the address DST
and the packet will be sent directly to this address. This connection will be successful only rarely, because the node having address DST is located usually out of the area of direct audibility.

The routing table contains the pairs (p)ath and (n)ode. The (p)ath represents the compared part of the address (here net) and (n)ode is address of node NN where leads the next packet step on it's way to DST address. The tables wide and local contains in (n)ode the lower address part only. The item (n)ode is the resultant information found in the routing tables and is inserted into address NN. More this situations can happen in the routing tables:

- In the local table there are not displayed the equal items, e.g. (p)ath 5E (n)ode 5E. This we can employ at removing of the unhelpful items in this way, that we put in (n)ode the same item like is in the (p)ath.

- It is not necessary to put in the local table the trivial path that is the ways to nodes, which are directly audible and which address is different in part "local" only. (In this case is "to"="dest" and (n)ode=(p)ath, see last note).

- In the tables global, net and wide are not displayed the items, where is (n)ode = 0. In this
way we can remove the unhelpful items by putting 0 in the item (n)ode.

- If the table doesn't contain the item (p)ath equal to compared part of address, then the packet is lost. The exception see next article.

- In the tables global and net there it is possible to use the item (p)ath, which is equal to respective part of address of node, in which the process runs (NODE ADDR). Then the relevant item (n)ode is the default address, where are sent the packets, which haven't own item (p)ath in the table.

The packet equipped now by the NN address goes to the next processing according to diagram "Block circuit of MORSE node". If the NN address equals to NODE ADDR, then the packet is sent to the link output L. The actual routing of the link output is written in the menu (N)ode (e)dit, in the column L. Some of serial channels or output to the Ethernet is used. The link output is used above all for the connection between two MCU working on different frequencies. The way of packets can go through nets with different frequencies and can be fully described in routing tables. On the other end of wire line there is next node, which accomplish the assignment of next packet step according to address DST, which the packet carries all time.

In next step is the address NN compared with addresses of other nodes in own CU. If the concordance is found, then the packet is sent through inner bus to the respective node. It is not necessary to have other setting in menu "Ne" or "SIe". In this node there begins the packet processing from start point again include using (different) set of routing tables.

Before sending to the net output the setting of flag "net to user addr" is checked. It is contained in the menu "Ne" in column M (Multiaddressing). If it is not set, which is the common case, then is the column M empty. By its setting the latter N appears here and then all
packets, originally going to the net output, are sent to the user output. The similar opportunity is to change routing of packets going originally to link retranslation output to the user output. Both options are used in special cases of routing only.

If the packet is no diverted to the link output nor to inner bus nor by multiaddressing, then it is sent to the net retranslation output N. It could be the radio or ethernet output assigned in menu "Ne" in column N.

Behind the network or link output of the node the output part of the access module continues in the connected communication channel. Here the packet (amongst other things) is again given a quadruple address. SRC and DST addresses remain unchanged, the value NN is inserted into the TO address, and the address of the node, in which the routing process took place, is (generally) written into the FRM address.

In this way the packet is sent to the net medium and for it's next processing takes charge this node, which address is equal to TO address contained in the packet.

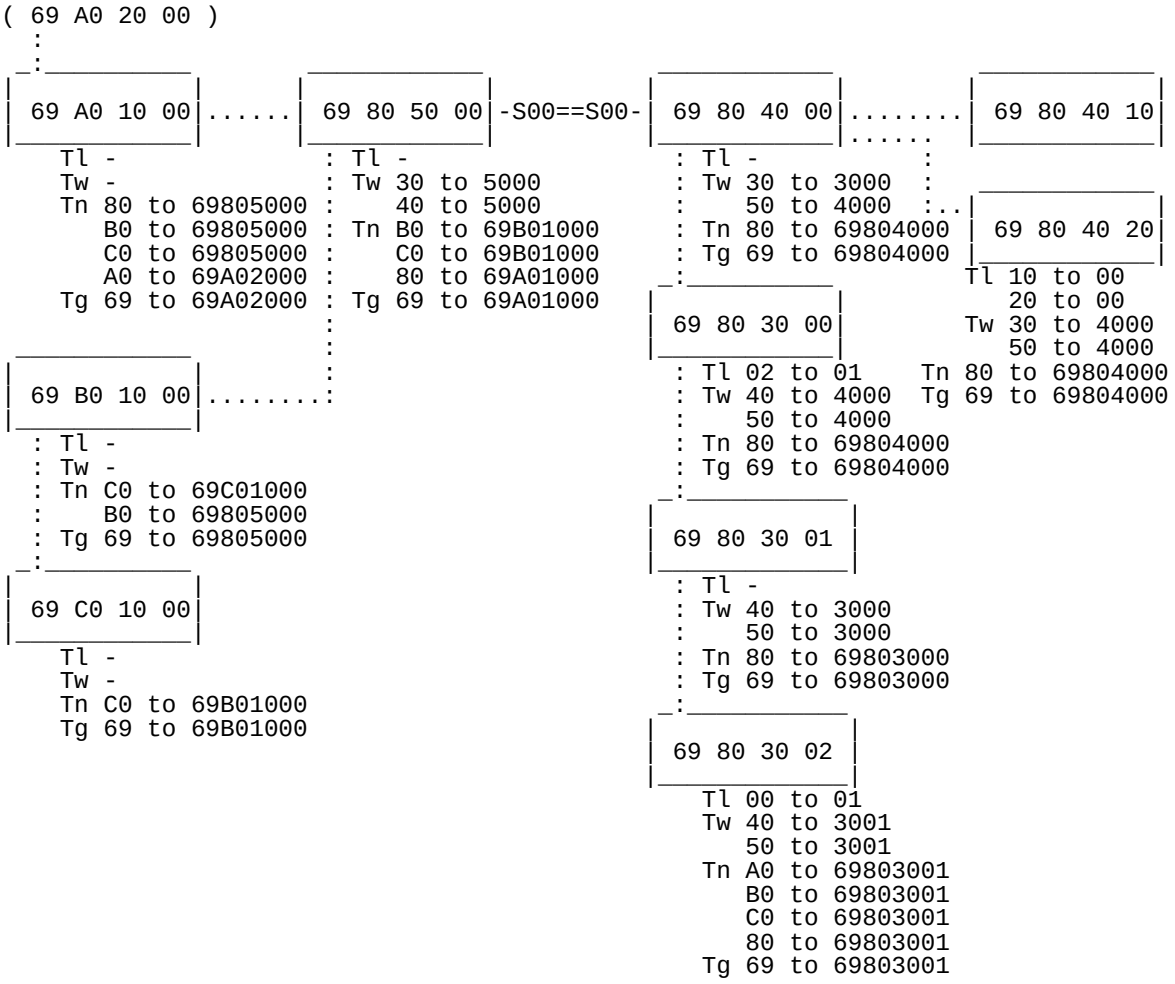
This description is the more detailed version of the article Routing 1.

Routing examples 4

More complicated network and the routing tables.

The network is made up of ten nodes which differ in addresses in terms of wide and net. The node 69805000 and 69804000 are connected by a serial link. The node 69A01000 is connected to other networks through node 69A02000 where routing continues.

It is good to remember the basic principle - each node seeks for the next hop according to the destination address, which is always contained in the packet. The node which receives the packet processes the packet in similar fashion.This process continues until the packet arrives at the destination.



Comments on the contents of the routing tables:

- The nodes 69804010 and 69804020 are contained in common CU. The necessary tables are almost identical - the only difference being in the local tables where node 69804010 needs the item 20 to 00 and node 69804020 needs the item 10 to 00. Both nodes use the common set of tables in (N)ode (e)dit.
- The node 69803002 contains all net address, which appear in the network in it's net tables. All these net addresses are pointed to node 69803001, so it is possible to use the default item 80 to 69803001, as was explained in the article Routing 2. The fact is used in the net and global tables of all other nodes.
- The node 69A01000 has it's default item in net table as A0 to 69A02000. i.e it's default route in net table is pointed to 69A02000, which connects our network to other networks. This is the reason the table contain explicit written net addresses which appear in our network even though they are all directed to a common node.
- At the wide table level, the default route option is not possible. This is why all wide table items which occur in the common net are written here. Similar reason is valid at the local table level as well, but this occur only rarely in our example.

The packet transfer is secured in this way; the receiving station checks the packet and sends an acknowledgement (ACK) about the error-free receiving of each packet. The transmitting station after obtaining this report regards the transfer to be completed. If it does not obtain ACK for the designated time (timeout) or if it obtains the no-acknowledge (NAK) report about wrong packet receiving, then it sends the packet again and this transmission is repeated pertinently until the number of repetitions is exhausted (repeat).

The security of transmission (to safeguard the packet) is pursued separately on every section of the route. The packet transmission between the user device and the SCC port is secured separately (according to attributes of protocol used) and the transmission on the individual sections of the MORSE route is also secured separately (according to the packet type setting).

The error free checking of packet transmitted is done in a way that the sending device does the given mathematical operation with the packet. The result, called checksum, is appended to the end of packet and then dispatch. The receiving device does the same operations with the packet and compares the result with the received checksum. If the checksums agree, then there is a high probability that an error didn't occur in the transmission and the message ACK is sent (e.g. at MARS-A protocol it is 0x06). If the checksums don't agree, then the message NAK is sent (at MARS-A 0x05).

Security in the user channel.

The transmission between the user device and the SCC channel is secured according to protocol used. Some protocols (e.g. async.link) do not secure transmission, for this reason they are suitable only for lines which are very reliable. Some protocols allow the security to be switched on/off in their parameters (RDS), and others have the security always on (MARS-A). For a successful transmission, the link must have on both ends the same protocol with the same parameters set.

Security in the net and link MORSE channel.

When the SCC channel receives the packet from outstanding device, it adjustes it in the unified format for the MORSE network. This format contains among others the byte `packet type` with bit `security` . The security system is activated according to this bit, which secures the transmitting in each step through the MORSE network.
The SCC channel behaves according to parameter `sec` in the menu `SIe` , which can get the values `ON, OFF, user` . These are set in the menu `SIe0u` by combination of parameters `user se(c)` and `(s)ecurity` .

```
Channel to Node Interface:
    retranslation     |   user                       lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1 ARt1             usr ON  NONE
(1) 0   NO AR         | 2 MASK 00000000/08 ON  OFF NONE
(2) 0   NO AR         | 1   NO AR          OFF OFF NONE
```

Influence of `sec` parameter to the packet type setting:

```
parameter sec v SIe:      security bit in packet type:
   ON                           ON
   OFF                          OFF
   usr   - according to packet type byte in incoming packet       ON or OFF
         - when incoming packet does not contain the packet type, it is set   ON
```

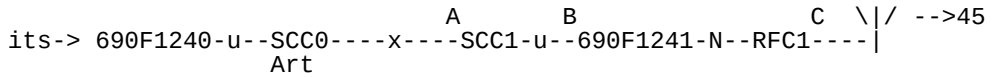
The security bit in the packet type (visible in the monitoring `iMS, iMF, iME`) then switches on/off the security at the radio transmission. The repeat parameters for it are prepared in the menu `FPe` in paragraph ACK:

```
RF channels:
    Access            |ACK             |coding |Mobile
 id a     del l num TO|fix var rep P hT|mod typ|net mask  center    per
(0) NORMAL 15 15  4 10| 600 400  5   30|REP DBL|OFF
(1) NORMAL 15 15  4 10| 600 400  5   30|REP DBL|OFF
(2) NORMAL 15 15  4 10| 600 400  5   30|REP DBL|OFF
(3) NORMAL 15 15  4 10| 600 400  5   30|REP DBL|OFF
(4) NORMAL 15 15  4 10| 600 400  5   30|REP DBL|OFF
```

The timeout is set by the sum `fix+(0to3)var` [ms], where `(0až3)` is the random number protecting against blocking, when two MCUs are sending concurrently. The max. number or repeats is set by parameter `rep` .

When the experimental packets are being sent, it is possible to put the packet type in the menu `its` in the item `(t)ype` . It is set 89 for user data secured or 09 for user data unsecured.

Next configuration is used for the ilustration:



The node 690F1240 simulates the user equipment called here terminal, which sends the packets to the node 690F1241 belonging to MORSE network.
From the menu "its" there is send to the node 40, the command which makes the node generate the packet and send it to the destination address 40. The node fulfills it and sends it at the next routing to it's user output, because the destination address is equal to it's own address.
The packet is given to the port SCC0, where is set the address changing by table Art1, which changes the destination address from 40 to 45. In the menu `SPe` there is set the protocol RDS, which parameters allows to check out several versions of security.

The packet addressed now to the destination 45 is given, by the crossed link, to the port SCC1. We follow the incoming packet in the point A by monitoring of SCC1. The packet MORSE is completed here and it is supplemented by packet type according to the parameter `SIe0u sec`. The resultant packet given to the node 690F1241 in the point B we can see by the iMSI monitoring.

The packet is sent from the node by channel RFC to the address 45. However it is out of reach and the ACK doesn't come. So we can see in the point C the work of repeated sending using the monitoring RFC.

The experiment we can try out on only one CU with next configuration (from main menu):

```
Ne 1a690f1240
   2a690f1241
   2l1
SIe 0uAt
    0ut1
AN1 d690F1245
    w690F1240
    p
iMSPe 1N
iMSIe 1ul100
      1uN
iMFPe l100
      N
its dL
    oL
    ahAAAA
    s
```

In the terminal and in the CU is the protocol RDS used, in which parameters is set the timeout `(a):1000ms` and number of repeats `(r):1`. Using the parameters
`(m)` - form of check sum and
`(c)` - switching ACK on/off we can simulate various situation:

Example 1.

Terminal generates the packets with different check sum, that is expected at CU, the ACK is off.

```
Terminal:                     Communication unit:
Parameters SCC0               Parameters SCC1
  RDS parameters:               RDS parameters:
  (a):1000  (r):1  (m):0000     (a):1000  (r):1  (m):FFFF
  (c) :OFF                      (c) :OFF

Monitoring in point A:

  10:16:28.125 rx;i    7 | S01
  4445 0200 AAAA 00
```

SCC1 in CU accepted the packet, checked it according to check sum as wrong and so SCC1 didn't create the MORSE packet and didn't send it next in the node.

Example 2.

The terminal generates again the packets with wrong crc but in this case expects the ACK.

```
Terminal:                     Communication unit:
Parameters SCC0               Parameters SCC1
  RDS parameters:               RDS parameters:
  (a):1000  (r):1  (m):0000     (a):1000  (r):1  (m):FFFF
  (c) :ON                       (c) :OFF

Monitoring:

  10:23:10.034 rx;i    7 | S01
  4445 0200 AAAA 00
  10:23:11.042 rx;i    7 | S01
  4445 0200 AAAA 00
```

SCC1 in CU evaluated the packet as wrong, but it has not permitted the ack sending, so it sends to the terminal no ACK nor refusing NAK. This is why the terminal after passing timeout 1000 ms once repeats the sending (it is set `(r):1`).

Example 3.

The wrong crc setting remains but the CU has permitted the ACK sending.

```
Terminal:                     Communication unit:
Parameters SCC0               Parametry SCC1
  RDS parameters:               RDS parameters:
  (a):1000  (r):1  (m):0000     (a):1000  (r):1  (m):FFFF
  (c) :ON                       (c) :ON

Monitoring:

  10:26:44.476 rx;i    7 | S01
  4445 0200 AAAA 00
  10:26:44.478 tx      1 | S01
  15
  10:26:44.492 rx;i    7 | S01
  4445 0200 AAAA 00
  10:26:44.494 tx      1 | S01
  15
```

SCC1 after receiving the wrong packet sent the reply = NAK = 0x15 and the terminal sent immediately the repeated packet. The reply on repeated packet was also NAK.

Example 4.

The terminal generates packet having the same crc as expects the CU. The parameter in CU in menu `SIe sec` is set to OFF.

```
Terminál:                     Communication unit:
Parametry SCC0                Parametry SCC1
  RDS parameters:               RDS parameters:
  (a):1000  (r):1  (m):FFFF     (a):1000  (r):1  (m):FFFF
  (c) :ON                       (c) :ON

Monitoring:

  10:36:24.538 rx;i    7 | S01
  4445 0200 AAAA 21
  10:36:24.542 tx      1 | S01
  06
  10:36:24.540|                 |690F1245 690F1241|S01I   OUT    2 09 0serv
  AAAA
  10:36:24.602|690F1245 690F1241|690F1245 690F1241|00A  RFTX     2*09 2dat
  AAAA
```

After receiving the packet, the SCC1 send back the ACK = 06, so the terminal doesn't repeat the sending. SCC1 created the MORSE packet from succesfuly checked packet received from wire link. According to the parameter `sec=OFF` in menu `SIe` the packet got the type 09 as we can see in CNI monitoring of port S01. The packet type 09 contains the information, that the packet will go through the MORSE network unsecured.
This is why the RF channel transmitted the packet to the address 690F1245 and in spite of it didn't become any ACK, the repeated transmission was not done.

We can see in the monitoring, that at creating the MORSE packet in the SCC1 channel was the data `AAAA` adopted from the original packet and the MORSE address 690F1245 was formulated from the short address 45. The others parts of original packet will be recreate in the SCC at the leaving the MORSE network and they are not transfered through the MORSE network.
Note - the second and the third item of monitoring was exchanged for better lucidity.

Example 5.

The parameters of link connection between the terminal and the CU remain the same as previously. In this case is in the SCC1 in menu `SIe` set the parameter `sec=ON`. In the menu `FPe` there are set the parameters for repeating on RF channel `fix=600, var=400, rep=2`.

```
Monitoring:

  11:06:55.874 rx;i    7 | S01
  4445 0200 AAAA 21
  11:06:55.877 tx      1 | S01
  06
  11:06:55.875|                 |690F1245 690F1241|S01I   OUT    2 89 0serv
  AAAA
  11:06:55.922|690F1245 690F1241|690F1245 690F1241|014  RFTX     2*89 4dat
  AAAA
  11:06:56.945|690F1245 690F1241|690F1245 690F1241|014  RFTX d   2*C9r4dat
  AAAA
  11:06:57.978|690F1245 690F1241|690F1245 690F1241|014  RFTX d   2*C9r4dat
  AAAA
```

From the error free packet incommed through the link into SCC1 was the MORSE packet created. According to the parameter `sec=ON` it was defined as type 89 (secured) and in this state it was send to the node 690F1241. The node transmitted it through the RF channel. The ACK doesn't come, so the transmitting was repeated twice more according to the parameter `rep` .

The same effect would have the parameter `sec=usr` in the menu `SIe`, because the packet incomming through the protocol having no security information, is in SCC defined as secured.
In other case, if in SCC would be used the MARS-A protocol carying the packet type information through the serial link, then the packet type defined in SCC1 would be done according to it.

Example 6 - complete secured transmitting.

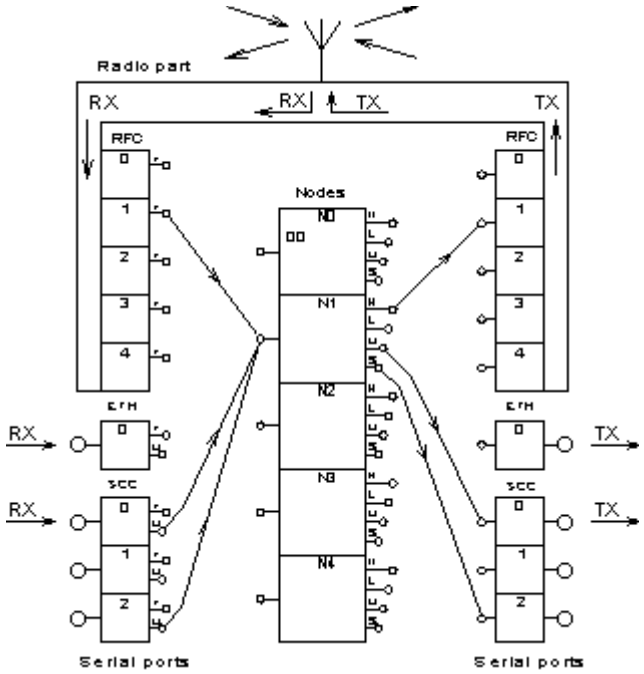
In the CU there was added the item to the routing table, which sends the packets having destination address 690F1245 to the node 690F1200. This node, whis is running near of our equipment, provides the ACK reply after receiving our MORSE packet.

```
Monitoring:

  11:43:07.368 rx;i    7 | S01
  4445 0200 AAAA 21
  11:43:07.372 tx      1 | S01
  06
  11:43:07.370|                 |690F1245 690F1241|S01I   OUT    2 89 0serv
  AAAA
  11:43:07.387|690F1200 690F1241|690F1245 690F1241|018  RFTX     2*89 0dat
  AAAA
  11:43:07.428|690F1241 690F1200|00000000 00000018|018*31~ 84    0|06  ack
```

We can see in the monitoring, that the packet `4445 0200 AAAA 21` obtained from the terminal was acknowledged by sending `06`. The packet was modified to the MORSE form having the packet type `89` and sent into node. From the node was the packet sent through RF channel to the address 690F1200 and the ACK was received.

By this actions was fulfilled the task of node 690F1241 and the next secured transmission of this packet is work for the next nodes.

Protocols



The CU diagram shows, that the packets from the user equipment comes into node through serial communication channel SCC0 or SCC1 or SCC2 (or ETH0). After passing through the MORSE network, they leave the last node again through SCC. Inside the MORSE network are used the packets in the MORSE form but the user equipment use packets in various forms. One of the tasks of SCC is to convert these packets into unified form suitable for the transfer through network and on its output adjust it again in users form. This function is defined in the protocols relevant to the various user equipment.

The protocols can be assigned individualy to each of SCC in the menu SPe, see chapter SCC channels for more details. Each protocol uses its set of parameters, which contain the user's information for the function of the protocol. The protocols can be classified, roughly, according to their function:

- Mere asynchronous link between two points. For example:
  ASYNC.LINK, ASYNC LINK++
- The protocols furnished by address and checksum. This allows the connection between different points of network, they guarantee the error free data transmittion. For example:
  RDS
- Protocols containing additional information useful in the network like packet type, packet number, notice about repeated transmitting or some special services in the link layer. For example:
  MARS-A
- Protocols MASTER-SLAVE type. This numerous group defines one station as the central, which sends the questions to other stations and expects the answers from them. For example:
  MDU, MININET, MODBUS, S-BUS, COMLI, PR2000
- Protocol comunicated through Ethernet channel.

Some illustrational examples for function of individual groups will be shown here. More exact descriptions are on the pages www.racom.cz in part Software, Protocols. Next examples shows the form of packet at input into the MORSE network through SCC having appropriate protocol, next the packet in MORSE form outputting the SCC to the node and at last the packet outputting the destination CU through channel SCC.

1. ASYNC LINK protocol

This simple protocol uses only one parameter - MORSE address of opposite station:

```
ASYNC LINK parameters:
link destination
(a)ddress:1241h
(q)uit
```

The next monitoring shows the packet containing the data AAAA, which was accepted by SCC channel S00. Next it was sent from SCC0 into the node modified in the MORSE packet form with addresses (from right) `source` and `destination`. The packet is processed in the node (addresses `from` and `to` added) and transmitted through RF channel and at last it leaves MORSE network in the destination station through channel SCC1 with the original form AAAA.

```
>>
Monitoring: source 690F120E|1.
07:33:59.013 rxsim   2 | S00
AAAA

CNI mon      |toa      frm     |dst      src      |          size|TT N
07:33:59.018|                  |690F1241 690F120E|S00I   OUT    2 89 0serv
AAAA

07:33:59.019|690F1240 690F120E|690F1241 690F120E|01E  RFTX     2*89 3dat
AAAA

Monitoring: source 690F1241|5.
07:33:59.105 tx      2 | S01
AAAA
```

The address of opposite CU must be put in the parameter of protocol ASYNC LINK. This is why the data can be transfered only between in advance chosen pair of CU (see the name of protocol). The data transferred between the user equipment and

SCC port are not checked on lost nor error free transfer.

2. RDS protocol

This protocol doesn't contain the address in its parameters, instead the address is contained in the incoming packet to SCC. It is the down part of destination address 41, more the packet contains the aid label 44, length of data 0200, checksum 78 and the data transfered AAAA. The data is the only part, which is transferred through the morse network, as we can see in second part of monitoring. The last part shows the packet leaving the MORSE network through SCC1 port.

```
Monitoring: source 690F120E|4.
13:44:09.670 rxsim   7 | S00
4441 0200 AAAA 25

13:44:09.684|                |690F1241 690F120E|S00I   OUT    2 89 0serv
AAAA

Monitoring: source 690F1241|3.
13:44:09.781 tx      7 | S01
440E 0200 AAAA 58
```

- One must say, that more information from input packet were transfered through MORSE network, in this case the address, which is the standard part of the MORSE packet. Transfer of this address through network in original input packet would be duplicated. Similarly duplicated would be the transfer of others part of input packet because it can be reconstructed by RDS protocol in output port SCC1.
- The comunication can be directed in different nodes, because the destination address is contained in the incoming packet. The user equipment should be able assemble the packet properly including the address and checksum.
- The checksum characterizes the state of packet at dispatch from the user equipment and according to it is checked the error free communication between outer equipment and the SCC port. In this illustrations are omitted the ACK messages, more about them in article Secured packet transfer .

```
RDS parameters:
(a):1000  (r):1  (m):FFFF
(c) :ON
(q)uit
>>
```

3. MARS-A protokol

The MARS-A protocol is equipped by all required functions. It contains in its head the address, the packet type, the repeated transmitting label, the numeration of packets transmitted and more. The example shows the dispatching of data AAAA to the address 690F1241. The second part shows the packet going from SCC to the node. The last one is the packet going out of the destination node through SCC1, which is adjusted in the MARS-A form.

```
>>
06:38:54.159 rxsim  12 | S00
C008 0980 690F 1241 AAAA 186C

06:38:54.173|                |690F1241 690F120E|S00I   OUT    2 89 0serv
AAAA

Monitoring: source 690F1241|2.
06:38:54.272 tx     12 | S01
C008 0900 690F 120E AAAA 1139
```

- The packet incoming into network through the port differs in its head and in checksum from the packet leaving the network, but the data is equal. The dissimilarity of head results from its function, e.g. the incoming packet contains the destination address, the outgoing packet contains the source address.
- The MARS-A protocol contains in its parameters mainly the timeout (a) and repeat (r). The others parameters are connected with other functions of protocol, which are explained in its describtion on www.racom.cz in chapter Software, Protocols.
- The target address is not written among the parameters but it is contained in the packet incoming in MARS-A protocol from outer device and so the packets from connected terminal can be send to different targets in the network.

```
MARS-A parameters:
(a):1000ms  (r):5  (c)rc:OFF  (l)oc:OFF  (G)SM:0  no traffic t(i)meout:0sec
(t)el.:
Opposite retranslation address:690F120F
(q)uit
>>
```

4. The data transfer between different protocols

Considering the fact, that the protocols converts the incoming packet to the unified MORSE form for transmission through the network and before outputing the packets converts again in the form required by outer device, it is possible to use defferent protocols on net input and net output. For example, the transport of data AAAA which was inputted by RDS protocol and outputted by MARS-A protocol looks like this:

```
>>
Monitoring: source 690F120E|6.
07:00:06.272 rxsim   7 | S00
4441 0200 AAAA 25

07:00:06.286|                |690F1241 690F120E|S00I   OUT    2 89 0serv
AAAA

Monitoring: source 690F1241|6.
07:00:06.362 tx     12 | S01
E008 0904 690F 120E AAAA D2FA
```

This feature can be used for creating the exchange, which proceeds the data from various CU equipped by different protocols.

5. MODBUS protocol as example of the Master-Slave protocol

This protocol is used for the communication among more members connected by wire network RS485. The MORSE version alows a radio part to be inserted in this network. In next example we can see the communication among the Master CU and two Slave CU addressed as 04 and 05 on RS485 network and the sending of data AAAAAAAA from Master to Slave 04. The Slave address is held in the first byte of the packet.

The basic run witout data transfer, the Master asks regularly all members of network:

```
15:28:06.780 tx      8 | S00
0403 1000 0001 809F             = Slave 4, do you have the data for me ?
15:28:06.784 rx;i    7 | S00
0403 0200 0074 44               = no
15:28:07.781 tx      8 | S00
0503 1000 0001 814E             = Slave 5, do you have the data for me ?
15:28:07.785 rx;i    7 | S00
0503 0200 0049 84               = no
15:28:08.782 tx      8 | S00
0403 1000 0001 809F             = 4, do you have ?
15:28:08.785 rx;i    7 | S00
0403 0200 0074 44s              = no
15:28:09.783 tx      8 | S00
0503 1000 0001 814E             = 5, do you have ?
15:28:09.787 rx;i    7 | S00
0503 0200 0049 84               = no
```

The protocol Modbus, running in the SCC0 port, sends the data to the Slave 04:

```
15:28:10.018|                |69000014 69000021|S00I    IN    409 6serv
AAAA AAAA                       = protocol Modbus Master got the data for Slave 04
15:28:10.784 tx      8 | S00
0403 2000 0001 8F9F             = Slave 4, do you have free the buffer ?
15:28:10.788 rx;i    7 | S00
0403 0200 0074 44               = yes
15:28:10.789 tx     21 | S00
0410 2000 0006 0C30 0A09 0669 0000 14AA AAAA AA4C A0     = save the message
15:28:10.794 rx;i    8 | S00
0410 2000 0006 4B9E             = I have saved it
```

The basic run continues:

```
15:28:11.785 tx      8 | S00
0503 1000 0001 814E             = Slave 5, do you have the data for me ? ...continues
15:28:11.788 rx;i    7 | S00
0503 0200 0049 84               = no
15:28:12.786 tx      8 | S00
0403 1000 0001 809F             = 4, do you have ?
15:28:12.790 rx;i    7 | S00
0403 0200 0074 44               = no
```

In standard Master-Slave protocol, all activities (transmitting or receiving) starts at the Master CU, which calls step by step the Slaves. The MORSE version of Modbus protocol offers more possibilities.

6. Ethernet protocol

The Ethernet protocol is used for the packet translation between MORSE network and IP network. This protocol can be used on CU having Ethernet port. The PPP protocol uses similar format and it is used on SCC port for connecting MORSE CU with IP router.

```
12:32:01.193 rx    2 | S00
BBBB

12:32:01.202|eth:TX   68 |0002A9 5841E9|0002A9 588C21| IP/UDP/MOR/RET/DAT
4500 0036 0010 0000 4011 F94C C0A8 0001 C0A8 0009 22B8 22B8 0022 0000 D200
0012 690F 8909 690F 8101 0B89 690F 8909 690F 8101 BBBB 415C
```

The Ethernet packet contains the IP header, UDP header and MORSE pseudoframe with data. The description see www.racom.cz, part MORSE Network, Beginner's Guide to MORSE.

Time in the Communication Unit (1).

Basic orientation in time is provided by the menu (s)ervice unit (s)tatus, which contains, amongst others, data about local time and GMT time:

```
local time:2003-04-03 07:52:39 LOC/daylight savings; secs from cold
start:947
  day time:2003-04-03 05:52:39 GMT; log write at:87213
```

These values are bound by the relationship:

```
GMT + DI(F)F + DST = LOC
```

where:
GMT is world time kept in the CU clock
DI(F)F is the time shift according to the geographical zone, e.g. for central Europe 3600 s
DST is a time shift of 1 hour during summer time (daylight savings)
LOC is the local time set in the menu (s)ervice (t)ime

Values DI(F)F and DST are controlled from the menu (U)nit (e)dit, which contains, but is not limited to:

```
Time (Z)one for DST:EU Time zone DI(F)F:3600sec     obsolete(<605) Time
(z)one:
69136sec
Summer time (all to zero - off):
start (1)month:3 (2)day:30
end   (3)month:10 (4)day:26
```

By selecting:
(Z)one (E)U
in this menu we switch on automatic use of summer time DST including annual automatic setting of the correct date at the beginning and end of DST. In this case the item "obsolete(<605) Time (z)one" is invalid.

However, by selecting:
(Z)one (N)ONE
we transfer to manual mode, where upon selecting items (1), (2), (3), (4) we choose the fixed date of the beginning and end of DST. If we insert zeros here, switching on of summer time is disabled.

Entry of the local time LOC is done in menu (s)ervice (t)ime:

```
Time:
(d)ay time   (s)econds
set (n)ew time
dat(e)  set ne(w) date
```

If we insert a date here which changes the year data it is then necessary to restart the CU (sgB) or carry out (I)nit and (S)ync in the "Ue" menu, or wait until midnight, when the year will update automatically.
The new local time entered using the commands "stw" and "std" is corrected according to the above-mentioned relationship by DI(F)F and DST and the GMT time obtained in this way is then maintained in the internal CU clock.

Example of transition to summer time.

Enter the time containing, for example, year 2004:

```
>>stw01.01.2004
```

In the "Ue" menu we set the automatic selection of the date for summer time (EU), local time shift of 3600sec and carry out Init and Sync:

```
Time (Z)one for DST:EU Time zone DI(F)F:3600sec

Ue Enter
I  Enter
S  Enter
```

Then in menu "Ue" we can read the dates of the beginning and end of summer time:

```
Time (Z)one for DST:EU Time zone DI(F)F:3600sec      obsolete(<605) Time
(z)one:69136sec
Summer time (all to zero - off):
```

```
start (1)month:3 (2)day:28
end   (3)month:10 (4)day:31

Enter the date and time just before the end of winter time:

>>stw28.03.2004
>>stn01:59:00

Now by repeatedly calling up menu "ss" we can monitor the transition to
summer time:

local time:2004-03-28 01:59:08 LOC; secs from cold start:19396
  day time:2004-03-28 00:59:08 GMT; log write at:105469

local time:2004-03-28 01:59:58 LOC; secs from cold start:19446
  day time:2004-03-28 00:59:58 GMT; log write at:105469

local time:2004-03-28 03:00:02 LOC/daylight savings; secs from cold
start:19450
  day time:2004-03-28 01:00:02 GMT; log write at:105469

Transition occurred at 01:00:00 GMT, when the difference between local time
and GMT increased by 1 hour.

Similarly, if we would like to monitor the transition from summer time to
winter time, we would have to set the date to 31.10.2004, local time to
01:59:00 and wait 61 minutes, until the local time reaches 03:00:00 (i.e.
01:00:00 GMT). Then the local time returns to 02:00:01 (i.e. 01:00:01 GMT).
```

Time in the Communication Unit (2).

Time in the MORSE CU is used in two formats:
- GMT - entries in the CU are stored and transmitted in this format
- local time - modified by the time zone and by summer time

GMT + DI(F)F + DST = LOC

where:
GMT is world time kept in the CU clock
DI(F)F is the time shift according to the geographical zone, e.g. for central Europe 3600 s
DST is a time shift of 1 hour during summer time (daylight savings)
LOC is the local time set in the menu (s)ervice (t)ime

Local time defined in such a way appears in the (s)ervice (t)ime menu, in the (s)ervice unit(s)tatus menu, in certain protocols and in older version it was used for "old (m)onitoring".

Local time, which appears in the message of the services (M)onitoring, stat (l)ogs and (E)vent log, obtained via Setr is the GMT time modified by the Setr time zone. This is determined when starting Setr by parameter -t, for example:

setr -t-7200 (corresponds to a time shift in Central Europe in the summer)

introduces a time shift of 2 hours from GMT. If the parameter is left out the value of the parameter saved in the setr.par file, which may be prepared in the same directory as setr.exe, is used. If the file setr.par is missing, a time shift value according to the host computer is used.

The result is that, for example, whilst monitoring the packet sent from the remote CU (in another time zone) to the local CU the time of sending and receiving differs at most by seconds. It is assumed that the clocks in both CUs are correctly synchronised.

Synchronisation of time in the MORSE system.

Precise sources of time can be:
- receiver of DCF time markers
- GPS system receiver
- sw MORCE running on Linux computer
- user protocols MARS-A, MODBUS, MAS, IEC 870-5-2(L&G) and others
- time can also be entered manually from the (s)ervice (t)ime

Synchronisation of time between neighbouring CUs can be set in the "Ue" menu:

Time sync:
 (i)n:1 Sr(c):690F5501 (p)eriod:3600
(O)ut:0 ds(t):00000000 p(e)riod:0

Data about time can be periodically taken from the neighbouring CU (1st row) or transferred to another CU (2nd row). The first method enables the synchronisation of more stations from one source.
The service is carried out by a node determined by selecting "(i)n" or "(O)ut", time data is requested from the "Sr(c)" CU or sent to the "ds(t)" CU, synchronisation is repeated with a period "(p)eriod" or "p(e)riod". By entering the periods (in seconds) the corresponding mode is activated.
The source or destination address needs to be selected as a directly accessible address, i.e. without routing. The accuracy of synchronisation depends on the density of radio traffic at the time of sending the packet, delays are in tens or hundreds of milliseconds.

During synchronisation the GMT time is transferred, in the synchronised CU the local time is then derived according to the "Ue" menu, in Setr according to parameter -t.

Monitoring.

Monitoring serves for monitoring packets passing through various MORSE network channels. Monitoring messages are generated by a selected node and sent to the selected address. This address can be local or can be one of the addresses in the MORSE network. The selection is made in the **ise** menu.

Monitored channels can be:

- SCC serial communication channels,
- RFC radio frequency channels
- or the ETH Ethernet channel.

Packets passing through have a different form on the outer (physical) layer of the channels, where they are modified according to the requirements of connected devices and different on the inner layer (CNI - channel to node interface), where they are modified to the standard form for communication between channels and nodes. This selection is made in the **iM** menu.

After setting up the configuration of monitoring and initialising it nothing immediately happens. However, the selected node monitors channels in the whole of its communication unit (CU) and as soon as a packet appears on any of the selected channels the node compiles a monitoring message in binary format and sends it through the MORSE network to the predetermined address. This address can also be local, however in this case the message is not transmitted through the network but via the service lead directly to the PC. The Setr application then modifies it into a readable format and displays it.

In the interests of restricting unnecessary transmission through the radio network SW filters are used which select only the required packets and only these are then monitored. Filtering runs according to addresses, according to the types of packets and according to other parameters.

Configuration of the method of message transmission:

In the `ise` menu it is possible to prepare up to 6 variations of message sending. For monitoring the use of rows `id(0)` and `id(1)` is recommended.

```
System channels: (Service 'iMo' works for s0 and s1 only)

id|--Node--addr------timeout---size---s(e)c--
(0)  0    0051877F     888     400     ON
(1)  1    690F4533     888     400     ON
(2)  0    0051877F     888     400     ON
(3)  0    0051877F     888     400     ON
(4)  0    0051877F     888     400     ON
(5)  0    0051877F     888     400     ON
```

For local monitoring the default setting is sufficient, see `(0)`. Here node 0 generates monitoring messages and sends them to its address, i.e. to its serial number `0051877F`. From here the default route (in the menu `(N)ode (e)dit`) to the service channel SCC2, to which it is connected the service lead. Another possibility is used in row `(1)`. Here node 1 generates the message and sends it to address `690F4533`. It is important that node 1 knows the route to this address according to the rules of routing.

The `timeout` and `size` items determine that the new message will not be sent sooner than 888 milliseconds or before at least 400 bytes of data for sending have gathered. The frequency of sending messages can be regulated by the selection of these parameters.

Item `s(e)c` determines that the monitoring messages will be transferred securely (the message will be repeated it the packet is lost).

Configuration of the selection of monitored channels:

Select the type of monitored channel in menu `iM` - SCC, RFC or ETH :

```
Monitoring:
(S)CC R(F)C (E)TH o(b)solete
(o)ff
(f)eatures
(q)uit
>>
```

In the next step at the selected channel the physical layer `(p)hysical` or operation between the channel and the node `CN(I)`is selected. The menu for the physical layer is different for each type of channel because they are designed for different environments. However, for CNI the menus are similar and only differ in the number of rows according to the number of channels because packets are already transferred in unified format between the channels and the nodes.

Monitoring SCC physical layers:

Here the situation is the most transparent, see the example. By selecting `iMSpe` we get to the menu for selecting one or more serial channels. Here in column `S` we then select the method of transferring monitoring messages for the selected SCC. For the above-mentioned configuration of the `ise` menu selection of s=0 causes messages to be sent to the locally connected PC, whereas when s=1 messages will be sent to address `690F4500`. Items `RX, TX, Ev` are for switching on monitoring of received or transmitted packets or events messages in the CU. The max. length of displayed data in the monitored packet is set using `len`, which has a default value of 32 bytes.

```
  SCC monitoring:

SCC--s---RX--TX--Ev----------len--------
(0)  0   OFF OFF OFF         32
(1)  0   OFF OFF OFF         32
(2)  0   OFF OFF OFF         32

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

Configuration of monitoring is completed by saving the configuration by means of `(w)rite` and `(I)nit`.

Particular care should be taken here. It is better to perform only `(I)nit` without saving using `(w)rite`, because this reduces the risk that monitoring inadvertently remains permanently on, unnecessarily loading the network. If `(w)rite` is not performed then monitoring will not start again upon restarting the CU at a later stage. In order to check that no monitoring is on, the `(s)ervice (s)tatus` command from the main menu comes in handy by submitting a message if monitoring is on. To switch off monitoring centrally the command `iMo`, selected from the main menu, is used.

Format of the monitoring message:

The header of the message contains time, packet direction tx/rx, packet length and the serial channel marking. Transferred data is given on the next row:

```
>>
07:14:11.377 tx     12 | S00
E008 0901 0000 0033 AAAA 4390
```

Monitoring of the Radio Channel.

Monitoring of the physical radio frequency channel enables monitoring of packets received and transmitted via the antenna. The antenna input/output is common for all five RF channels, which are not more distinguished here. Owing to the large number of captured packets it is useful to use filtering here for selection of the required packets.

From the main menu choose `iMFpe`

```
RFC monitoring:
o(N) o(F)f          (R)X:OFF  (T)X:OFF  Medi(u)m:OFF
(s)ys. channel:0
(l)ength:0     rx (p)romisc. lvl:OFF
Filter:
Only (h)ead crc OK :OFF
Only d(a)ta crc OK :OFF
Packet t(y)pe:0000 tmas(k):0000
(d)st :00000000  (D)st mask :00000000 (1)-for :RX
sr(c) :00000000  sr(C) mask :00000000 (2)-for :RX
t(o)  :00000000  t(O) mask  :00000000 (3)-for :RX
fro(m):00000000  fro(M) mask:00000000 (4)-for :RX
($)dsc: 7817
de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

Monitoring is switched on and off using commands `(R)X:OFF` and `(T)X:OFF` for individual directions or commands `o(N), o(F)f` for `RX` and `TX` simultaneously. Assignment of the recipient of monitoring packets is again done by assigned the system channel `0` or `1` . The length of the displayed data is contained in the `(l)ength` parameter. These settings are sufficient for the basic RF channel monitoring function.

Filtering packets in monitoring.

The RFC monitoring menu contains filters, which can be used to suppress the monitoring of unwanted packets. If the filter is set, for example:

```
Packet t(y)pe:0089 tmas(k):00FF
```

only type `89` packets, i.e. user packets, are monitored. The other packets, e.g. service or ack, are not monitored.

Packets can also be filtered according to their addresses. For example, by writing:

```
(d)st :690F4500  (D)st mask :FFFFFF00 (1)-for :RX
```

only received packets whose destination address is in the range `690F4500` to `690F45FF` will be monitored.

The RFC monitoring menu contains 7 different filters. The received packet must meet the requirements of all switched on filters in order for it to be monitored.

Examples of monitoring the SCC - Local.

Example 1.

We shall start from default states. Local monitoring of SCC0 is switched on from the main menu as follows:

```
 iMSpe    Enter Enter  - monitoring SCC, physical layer
      0n  Enter Enter  - common switching of RX and TX for SCC0
```

The result is as follows:

```
  SCC monitoring:

SCC--s---RX--TX--Ev----------len--------
(0)  0   ON  ON  OFF         32
(1)  0   OFF OFF OFF         32
(2)  0   OFF OFF OFF         32

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

Perform initialisation:

```
 I  Enter Enter
```

Next we create a packet. We use the `(t)est` function, which generates a packet in the node and sends it to the entered address. From the main menu:

```
 its    Enter       - select function (s)end packet
 dL     Enter Enter - enter local address
 ahaaaa Enter Enter - enter hexadecimal characters AAAA
                      into item d(a)ta
```

Appearance of the resulting menu:

```
Send packet: (N):1   (d):690F5501h
(t)ype:0009h
s(o)urce:690F5501h
d(a)ta:..
random data (l)ength:0byte
(r)epeat period:0ms + (j)itter:0ms
IP (P)ing
(s)tart r(e)port sto(p)
(q)uit
>>
```

After sending the test packet

```
 s      Enter
```

an SCC monitoring report appears:

```
>>
10:07:30.232 tx     12 | S00
E008 0906 0000 0001 AAAA 43A5
10:07:31.240 tx     12 | S00
E808 0906 0000 0001 AAAA 4BA5
10:07:32.249 tx     12 | S00
E808 0906 0000 0001 AAAA 4BA5
10:07:33.258 tx     12 | S00
E808 0906 0000 0001 AAAA 4BA5
10:07:34.267 tx     12 | S00
E808 0906 0000 0001 AAAA 4BA5
10:07:35.276 tx     12 | S00
E808 0906 0000 0001 AAAA 4BA5
```

Description of the process - test `(s)end packet` orders node 1 to send data `AAAA` to address `690F5501` . This address is its own address and that is why data was transferred to the default configured channel SCC0, containing protocol MARS-A. The packet, modified into MARS-A format, which is sent from SCC0 to the external user, is seen in the monitoring data row. Owing to the fact that there is nothing connected to the SCC0 output the protocol does not receive confirmation about receipt of the packet and therefore repeats transmission 5 times.

Example 2.

Add the address of the next node, `690F5505` , to the CU and replace the MARS-A protocol with a simple asynchronous link protocol. In the next description Enter, write and initialisation commands are omitted.

```
 Ne
 2a690F5505   - enter new address

 SPe 0oa      - select async. link protocol on SCC0
     0t 5505  - the parameter is the address of target node 2
     1oa      - select async. link protocol on SCC1
     1t 5501  - the parameter is the address of target node 1
```

The resulting configuration can be diagrammatically represented as follows:

```
  as.link..SCC0..u..690F5501
                   690F5505..u..SCC1..as.link
```

Switch on monitoring of SCC0 and SCC1:

```
 iMSpe 0n
       1n
```

We send data `BBBB` to address `690F5505` using the `(s)end packet` function:

```
 its d690F5505
     ahBBBB
     s

>>
11:14:15.576 tx      2 | S01
BBBB
```

The report says that in a certain time 2 bytes were sent from SCC1. The packet of the async. link protocol has no header or checksum and only contains the transmitted data `BBBB`. It does not even use confirmation of receipt and therefore data is only sent once.

The test packet can also be entered with the `channel se(n)d` function at the SCC0 input:

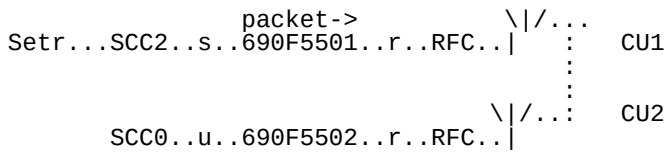
```
  in
  ahCCCC
  s

>>
11:14:51.344 rxsim   2 | S00
CCCC
11:14:51.358 tx      2 | S01
CCCC
```

Monitoring shows a simulated packet `rxsim` entering SCC0, which then leaves `tx` from SCC1 after passage through both nodes.

Examples of monitoring the SCC - Remote.

Example 1.

In practice verification of operation on the SCC port of a remote CU is common. An example of configuration:



Creation of this configuration from the default state:

```
CU2:
 Ne 1a690F5502  - local entering of address

CU1:
 Ne 1a690F5501  - local entering of address
```

Test the connection from CU1 to CU2:

```
 !h02
 !

u S02   690F5501    R01
31/ 66  690F5502   serd

 serd   690F5502    R01
30/ 67  690F5501  u S02
690F5502h>
```

During remote configuration proceed in the same way as for local configuration. The main difference is that after commands `(e)dit, (w)rite, (I)nit` we should wait until the message `get, write, put` appears informing us that information from the remote CU has arrived to our PC. Only then do we press `Enter` to display this information.

Monitoring in CU2 is switched on remotely:

```
 ise Enter  - command for editing system channels

690F5502h>
get  SYSCH0 O.K.
get  SYSCH1 O.K.
get  SYSCH2 O.K.
get  SYSCH3 O.K.
get  SYSCH4 O.K.
get  SYSCH5 O.K.
690F5502h>
```

Wait until this message arrives from CU2 and then press Enter:

```
System channels:
(Service 'iMo' works for s0 and s1 only)

id|--Node--addr------timeout---size---s(e)c--
(0)  0    004C9D8F     888     400     ON
(1)  0    004C9D8F     888     400     ON
(2)  0    004C9D8F     888     400     ON
(3)  0    004C9D8F     888     400     ON
(4)  0    004C9D8F     888     400     ON
(5)  0    004C9D8F     888     400     ON

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F5502>

 1N1        Enter Enter  - in the system channel, s1,
                           N1 is defined as the monitoring node
 1a690f5501 Enter Enter  - in the system channel, s1, 690F5501
                           is defined as the target address
                           for the monitoring message
System channels:
(Service 'iMo' works for s0 and s1 only)
```

```
id|--Node--addr------timeout---size---s(e)c--
(0)  0    004C9D8F     888     400     ON
(1)  1    690F5501     888     400     ON
(2)  0    004C9D8F     888     400     ON
(3)  0    004C9D8F     888     400     ON
(4)  0    004C9D8F     888     400     ON
(5)  0    004C9D8F     888     400     ON

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F5502h>
```

We send this resulting version of the menu to be written into CU2:

```
 w Enter

690F5502h>
write SYSCH 0 O.K.
write SYSCH 1 O.K.
write SYSCH 2 O.K.
write SYSCH 3 O.K.
write SYSCH 4 O.K.
write SYSCH 5 O.K.
690F5502h>I
```

When this message appears we perform Init in CU2:

```
 I Enter

690F5502h>
put   SYSCH 0 O.K.
put   SYSCH 1 O.K.
put   SYSCH 2 O.K.
put   SYSCH 3 O.K.
put   SYSCH 4 O.K.
put   SYSCH 5 O.K.
690F5502h>
```

Now we return to the main menu in CU2 and open the menu for monitoring:

```
 Q     Enter Enter
 iMSpe Enter

690F5502h>
get  MON SCC0 O.K.
get  MON SCC1 O.K.
get  MON SCC2 O.K.
690F5502h>
```

After this message press Enter:

```
  SCC monitoring:

SCC--s---RX--TX--Ev----------len--------
(0)  0   OFF OFF OFF         32
(1)  0   OFF OFF OFF         32
(2)  0   OFF OFF OFF         32

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F5502h>

 0s1 Enter Enter  - for SCC0 monitoring we designate
                    s1 as the system channel,
                    which is directed to CU1
 0n  Enter Enter  - switch on SCC0 monitoring for the
                    transmitted and received packets

  SCC monitoring:

SCC--s---RX--TX--Ev----------len--------
(0)  1   ON  ON  OFF         32
(1)  0   OFF OFF OFF         32
(2)  0   OFF OFF OFF         32

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F5502h>
```

We send this resulting form of the menu to CU2
for Initialisation:

```
 I Enter  - Initialisation in CU2

690F5502h>
put   MON SCC 0 O.K.
put   MON SCC 1 O.K.
put   MON SCC 2 O.K.
690F5502h>

 Enter
```

It is also possible to write into flash memory in CU2 using the command w Enter. However it is more suitable to omit this permanent entry. A potential stray monitoring will not start again after restarting the CU and will not load the network.

From this moment monitoring messages about operation on SCC0 are sent from CU2 to CU1. If there is no operation on SCC0 we can send a packet there from CU1:

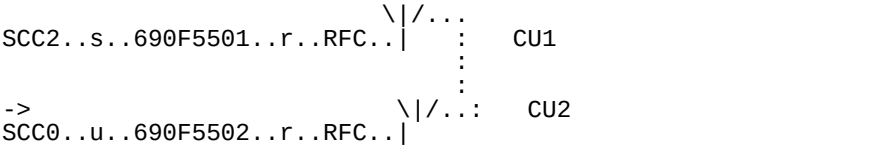
```
 Q         - to main menu
 !l        - return to CU1
 its       - menu (s)end packet
 d690F5502 - target address for sending packet
 ahAAAA    - data for sending AAAA

Send packet: (N):1   (d):690F5502h
(t)ype:0009h
s(o)urce:690F5501h
d(a)ta:..
random data (l)ength:0byte
(r)epeat period:0ms + (j)itter:0ms
IP (P)ing
(s)tart r(e)port sto(p)
(q)uit
>>
 s         - start, odeslání paketu

>>
Monitoring: source 690F5502|1.
09:15:58.855 tx     12 | S00
D008 0900 0000 0001 AAAA 73A3
09:15:59.864 tx     12 | S00
D808 0900 0000 0001 AAAA 7BA3
09:16:00.873 tx     12 | S00
D808 0900 0000 0001 AAAA 7BA3
09:16:01.883 tx     12 | S00
D808 0900 0000 0001 AAAA 7BA3
09:16:02.892 tx     12 | S00
D808 0900 0000 0001 AAAA 7BA3
09:16:03.901 tx     12 | S00
D808 0900 0000 0001 AAAA 7BA3
```

The received monitoring message documents the operation of port SCC0 in the CU2. Here it is protocol MARS-A, and that is why data AAAA is supplied with the respective header and checksum. A user device is not connected that would confirm receipt, therefore transmitting is repeated five times.

Example 2.

Monitoring packet entering the SCC in the remote CU.

For testing the passage of a packet from port CU2 to CU1 it is necessary to enter a packet into SCC0 in CU2 with the form required by the installed protocol. Composition of the MARS-A packet is more demanding, therefore for this purpose we set a simple asynchronous link protocol to SCC0 in CU1 and also to SCC0 in CU2:




```
 >>SPe 0oa    - enter async. link protocol to SCC0 in CU1
       0t a02 - the parameter is the last byte 02 of the CU2 address
 iMSpe 0n     - switching on monitoring SCC0 in CU1

 !h02         - switching over to CU2
 !            - testing connection
 SPe 0oa      - enter async. link protocol to SCC0 in CU2
     0t a01   - the parameter is the last byte 01 of the CU1 address
```

```
 iMSpe 0n     - switching on monitoring SCC0 in CU2
 in ahBBBB    - menu channel se(n)d, data BBBB
    r5000     - interval of repeated transmission 5sec

Resulting form of the menu:

Channel data send.
(d)estination: SCC0-RX
d(a)ta :..
(r)epeat period :5000ms
(s)tart sto(p)
(q)uit
690F5502h>

    s  - start of packet sending

Monitoring message in the CU1:

Monitoring: source 0051877F|4.
12:31:35.657 tx      2 | S00
BBBB
Monitoring: source 690F5502|5.
12:31:35.098 rxsim   2 | S00
BBBB
Monitoring: source 0051877F|5.
12:31:40.657 tx      2 | S00
BBBB
Monitoring: source 690F5502|6.
12:31:40.098 rxsim   2 | S00
BBBB

    p  - stop sending
```

The route of the packet from the SCC of the remote CU2 to the SCC of the local CU1 is verified in such a way.

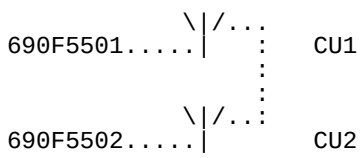
Monitoring contains a pair of entries from node `690F5502` (CU2), where the packet enters the MORSE network and from node `0051877F` (node 0 in CU1), where it leaves the network. These two entries have their sequence switched over here, which is caused by the longer route of the message from CU2. The mutual time sequence of these entries is alright here (the difference between tx and rx is 559ms). This value, however, is not conclusive because monitoring is marked by the time of the CU in which it occurred and mutual synchronisation between the CU is often less precise than the speed of the monitored processes.

Example of Monitoring the Radio Channel.

Example 1.

Local monitoring of the physical layer of the RFC.

Prepare the radio link in the following configuration:



From the default state it means this setting (including the relevant saving w, I):

```
CU2:
  Ne    1a 690F5502 - entering address
CU1:
  Ne    1a 690F5501 - entering address
  iMFpe l100        - max. length of monitored data
        N           - switching on monitoring
```

In node 690F5501 we generate a packet with data AAAA and send it to address 690F5502:

```
  its               - test Send packet
        d690F5502   - target address
        ahAAAA      - transmitted data AAAA
        s           - start
```

This message appears:

```
>>
Monitoring: source 0051877F|1.
RF mon      |toa      frm     |dst      src     |lNo!DQ!RSS size|TT N
11:16:56.907|690F5502 690F5501|690F5502 690F5501|058  RFTX     2*09 0dat
AAAA
```

Composition of the message:

```
Monitoring: source 0051877F|1.
```

The source of the monitoring message; here it is the address of node 0 in CU1. This row only appears when changing the generator of the message, i.e. when switching messages from different CUs.

```
RF mon      |toa      frm     |dst      src     |lNo!DQ!RSS size|TT N
```

The header for marking items in monitoring rows. The order of the addresses (from right - `source, destination, from, to`), which are contained in the message, is important. This header only appears once, during the first monitoring after starting Setr, to save space.

```
11:16:56.907|690F5502 690F5501|690F5502 690F5501|058  RFTX     2*09 0dat
AAAA
```

The actual message which says that the packet with data `AAAA` was sent from the `src` address `690F5501` to the `dst` target address `690F5502`. In this case the link is merely composed of two stations, therefore the pair of addresses `frm` and `to` are the same as addresses `src` and `dst`. In place of the quality and strength of the signal, DQ and RSS, the RFTX label is there during transmitting. The length of data size is 2 bytes, the type of packet `TT` is `09` , which means unsecured packet transmission.

Example 2.

The influence of packet type to the monitoring.

In the `its` menu we change the type of transmitted packet to secured, i.e. `89` :

```
 its t89
```

The resulting form of the menu:

```
Send packet: (N):1   (d):690F5502h
(t)ype:0089h
s(o)urce:690F5501h
d(a)ta:..
random data (l)engh:0byte
(r)epeat period:0ms + (j)itter:0ms
IP (P)ing
(s)tart r(e)port sto(p)
(q)uit
>>
```

using the `(s)tart` command we send the packet and receive a message:

```
>>
11:17:23.003|690F5502 690F5501|690F5502 690F5501|059  RFTX     2*89 3dat
AAAA
11:17:23.044|690F5501 690F5502|00000000 00000059|059*28~ 64    0|06  ack
```

The packet is transmitted as secured, see packet type `89` at the end of the first row. Thus an `ack` acknowledgement was received from the receiving CU2. Ack is a matter of RF channels of the pair of CUs, which are functioning as `from` and `to`. Therefore only addresses `frm` and `to` are used. Addresses `src` and `dst` in the ack message have no sense even though a MORSE address can formally appear at these points.

Example 3.

Filtering according to the type of packet.

During transmission of monitoring messages through the radio channel it is important to omit monitoring of the service (i.e. monitoring) packets so that there is no cycling of transmission. We therefore set the filter to select only type `89` packets, i.e. secured user packets. In the last example this will result in the repeated disappearance of the `ack` message.

Setting the filter:

```
  iMFpe y0089
        k00FF
```

After initialisation, I, and reading, S, we see that mask `00FF` has changed to `00DF`. This is because the respective (11th) bit of the type of packet can take on different values and therefore the mask avoids it.

```
RFC monitoring:
o(N) o(F)f          (R)X:ON  (T)X:ON  Medi(u)m:OFF
(s)ys. channel:0
(l)ength:100     rx (p)romisc. lvl:OFF
Filter:
Only (h)ead crc OK :OFF
Only d(a)ta crc OK :OFF
Packet t(y)pe:0089 tmas(k):00DF
(d)st :00000000  (D)st mask :00000000 (1)-for :RX
sr(c) :00000000  sr(C) mask :00000000 (2)-for :RX
t(o)  :00000000  t(O) mask  :00000000 (3)-for :RX
fro(m):00000000  fro(M) mask:00000000 (4)-for :RX
($)dsc: 7817
de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>
```

After transmitting the packet

```
  its s
```

we receive a message containing only a type `89` packet:

```
>>
13:56:37.069|690F5502 690F5501|690F5502 690F5501|065  RFTX     2*89 0dat
AAAA
```

Example 4.

Monitoring of the remote CU.

We connect remotely to CU2 and test the connection.

```
 !h02 Enter
 !    Enter
```

In CU2 switch on monitoring directed towards node 690F5501 in CU1:

```
 ise   1N1         - in system channel 1 it monitors node 1
       1a690F5501  - message sent to address 690F5501
 iMFpe s1          - send according to the setting in system channel 1
       l100        - length of data in monitored packet
       y89         - monitor only user type packets
       kff         - mask for type of packet
       N           - switch on RX and TX direction
```

The resulting form of the menu in CU2:

```
RFC monitoring:
o(N) o(F)f          (R)X:ON  (T)X:ON  Medi(u)m:OFF
(s)ys. channel:1
(l)ength:100     rx (p)romisc. lvl:OFF
Filter:
Only (h)ead crc OK :OFF
Only d(a)ta crc OK :OFF
Packet t(y)pe:0089 tmas(k):00DF
(d)st :00000000  (D)st mask :00000000 (1)-for :RX
sr(c) :00000000  sr(C) mask :00000000 (2)-for :RX
t(o)  :00000000  t(O) mask  :00000000 (3)-for :RX
fro(m):00000000  fro(M) mask:00000000 (4)-for :RX
($)dsc: 7817
de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
690F5502h>
```

In CU1 prepare the menu test for the packet transmitting:

```
 its d690F5502  - destination address
     t89        - packet type user
     ahAAAA     - data sent

Send packet: (N):1   (d):690F5502h
(t)ype:0089h
s(o)urce:690F5501h
d(a)ta:..
random data (l)ength:0byte
(r)epeat period:0ms + (j)itter:0ms
IP (P)ing
(s)tart r(e)port sto(p)
(q)uit
>>

     s          - start
```

After sending the packet we get the monitoring message from CU2 about receiving the packet by RF channel:

```
>>
08:33:10.602|690F5502 690F5501|690F5502 690F5501|01D*30* 47    2|89 0dat
AAAA
```

The report says, that the packet of type `89` containing the data `AAAA` comms from the address `690F5501 (src and from)` to the address `690F5502 (dst and to)`. At the same time the packets carrying this message are not monitored, because they have different packet type, then `89` .

If this packets would not filtered off, then the cykling of monitoring occurs and always additional packets would be sent. In this situation, when the screen is full of new messages, we can stop the monitoring in source CU using commands:

```
 Q   Enter  - return in the main menu
 iMo Enter  - all monitorings stop
```

This commands can be put "in blind" regardless other messages appears between it's characters. The putting of iMo can be repeated in case of failure.

Ethernet in MORSE.

The Ethernet channel in the MORSE CU offers 3 modes of operation, M-IP-M, IP-M-IP, MAS:

*******   M-IP-M   *******

The packet travelling through the MORSE network has a section going along the IP network inserted into its path. The packet has the following configuration:

```
   MORSE       ->            IP            ->    MORSE
|MORSE packet| -> |IP header|MORSE packet| -> |MORSE packet|
```

Looking at the configuration of the MORSE network the IP section appears as one of the routing jumps between two nodes. Routing tables are compiled in the same way as if this section would be one of the radio jumps. The relationship with IP is configured in the Ethernet channel and Art tables in two CUs at the interface with the IP part.

*******   IP-M-IP   *******

The packet in the IP network has a section going along the MORSE network inserted into its path. Packet configuration:

```
   IP       ->            MORSE         ->    IP
|IP packet| -> |MORSE header|IP packet| -> |IP packet|
```

For the MORSE network the event begins with receipt of the packet from the user output of the Ethernet channel to the node and ends with transfer of the packet through the user output of the last node to the Ethernet channel. The relationships with IP are contained in the Ethernet channel and Art tables in the end CU.

*******   MAS   *******

The MORSE Application Server is designed for connecting an application running on the LAN network with end points lying in the MORSE network. MAS replaces the IP header with the MORSE header and vice versa:

```
            MORSE         <-->          IP
     |MORSE header|data|  <-->  |IP header|data|
```

If the application in the LAN network sends a UDP datagram to any of the terminals in the MORSE network then the MORSE destination address is contained in the data of this datagram and is used by MAS to compile the MORSE header. The "source" MORSE address as well as the second main parameter of the MORSE header is derived from the senders IP address and port by means of Art. In a similar manner the IP destination address and port number are derived from the MORSE destination address when processing the incoming packet from the MORSE network. The MORSE source address is then inserted into the MORSE pseudoframe in the data part of the UDP datagram.

A description of these modes is given in the Ethernet chapter of MORSE firmware documentation.

Connection via MORSE Application Server (MAS).

Situation - connection of AAA application in Windows through MAS with CU (Communication Unit) MR25ET and to the MORSE network. The AAA application is replaced here by Setr.

(MCM302ET (ECO board), MR900 with Ethernet module or sw morce work in the same way as MR25ET.)

Configuration consists of the following steps:

```
1) Entering IP addresses - into PC and through service cable to CU
2) Ping testing          - ping PC -> CU along Ethernet link
3) Starting Setr         - CU connected via Ethernet link, ping PC <- CU
4) Setting routing       - configuration for a single PC or for more users
5) Setting MAS           - select range of PC addresses, filling Art table
6) Test from mtu menu    - sending UDP datagram from Setr via MAS to remote CU,
                           sending MORSE packet from CU via MAS to PC
7) Starting application  - after this activation simply replace Setr with
                           another AAA application and the connection is ready
```

The numbered steps are described below in detail, with a brief summary given at the end.

(1) Local connection via Ethernet

First set a suitable IP address and mask in a PC running Windows98, for example:

```
  IP address - 10.0.0.1
  mask     - 255.255.255.0
```

The MORSE CU of the MR25ET is in its default state. Connect to the CU using the service cable and set the IP address and mask, e.g.:

```
  Epe i0A000002    ... i.e. 10.0.0.2
      nFFFFFF00    ... i.e. 255.255.255.0
```

(2) Now we can remove the service cable and connect the PC and CU using the crossed Ethernet cable. Test the connection by sending a ping from the command line in the PC to the MR25ET:

```
  ping 10.0.0.2
```

(3) If the ping works we can start Setr in the PC running Windows:

```
  setr -pIP10.0.0.2
```

We can test this PC <-> CU connection in local mode by calling certain services, e.g.:

```
  sts    ... which replies with the number of seconds since starting the CU,
e.g.:
  >>312
```

Now it is possible to test a ping from the CU to the PC:

```
  Epe tP t0A000001   ... IP address of PC
         s           ... (s)tart
         e           ... r(e)port gives the result
         p           ... sto(p)
```

If no reply returns we should search for the error in the steps carried out so far.

Setr connected in such a way is only able to communicate with the local CU. Connection with other MORSE network CUs is possible after configuring MAS, see below.

Note:
If we are working in LAN with the name server in which the CU address is defined we can enter the name of the computer instead of the IP address when starting Setr, e.g.:

```
  setr -pIPradiomodem.racom.cz
  setr -pIPradiomodem
```

(4) Connection via MAS

Now we have to decide if we will use only one single MORSE address and one single application, or a group of 256 addresses and more applications (on more computers).

(4)(A) Single address.

Up until now the MORSE address was not required in the CU. Now we can insert the chosen address:

```
  Ne 1a 690F5600  ... MORSE address
     1uE0         ... routing the user output from the node to the AAA
application
     1sE0         ... routing service packets to Setr
```

Set the EIe menu to the default condition:

```
  EIe f     ... routing the user output of the Ethernet channel to node 1
```

(5)(A) Setting the Epe menu for MAS mode:

```
  Epet t s1          ... link to Art table number 1
         B690F5600   ... Base - required PC MORSE address
         M00000000   ... zero mask requires an absolute match for the Base and
PC addresses
```

Now it is necessary to fill in Art table number 1. We have the possibility of doing this automatically or manually using Setr. For automatic filling switch off Setr (Alt+X) and then start it again using command:

```
  setr -pIP10.0.0.2 -pm8000 -pw690F5600
```

where the meaning of parameters is:

```
  -pIP10.0.0.2   ... IP address of CU
  -pm8000        ... number of UDP port of application (here Setr), in decimal
  -pw690F5600    ... MORSE address, under which the application will be known in
the MORSE network, in hexadecimal
```

After starting Setr read the contents of the Art table:

```
  A r
>>
ART No 1:
items: 1
default gw: 00000000 (0.0.0.0        )
dest:    gw:
690F5600 1F400001 (105.15.86.0     31.64.0.1      )
>>
```

The Art table contains:
in the "dest" column the MORSE address, under which application AAA is known in the MORSE network,
in the "gw" column there is a composite expression - let´s note that 0x1F40 is 8000 in decimal, which is the UDP port of our AAA application and furthermore 0x0001 is the lower part of the IP address of the computer running windows, from which only the lower 14 bits are valid and the upper 2 bits are zero.

We can fill the Art table in this manually for the necessary combination of addresses. It is useful to block automatic filling with the command:

```
  Epe t ef    ... (e)nable:OFF
```

Appearance of the respective part of the menu:

```
MAS:
  (s)Art:1; write (e)nable:OFF
  (B)ase:690F5600  (M)ask:00000000
```

Now MAS is ready to transfer UDP datagrams from the AAA application, operating at the IP address 10.0.0.1 and at port 8000,
which appears as address 690F5600 in the MORSE network.

We can check functionality using Setr. Switch Setr off and start with the parameters:

```
  setr -pIP10.0.0.2 -pm8000 -pw690F5600        or
  setr -pI0A000002 -pm0x1F40 -pw690F5600

  >>!h
  690f5600>sts
  690F5600>1245  ... example of a reply
```

If no reply arrives search for the error in the steps according to points (4) and (5).

Note:
If our AAA application uses various sockets (different UDP ports) for receiving

and for transmitting, add the following row to the Art table manually:

```
  690f5600 1F400001
  00000000 1F400001
```
(Caution, the order of these rows is important)

This means that the mode will accept packets from any UDP port the same as if it had come from port 8000. This definition of the default MORSE address for the application can also be used in mode ad B), but only for one of the applications.

(4)(B) More addresses, connecting Setr via LAN.

Situation - The MR25ET and several PCs are connected to the LAN network. Through the common MR25ET, on which MAS is running, it is possible to connect these PCs to the MORSE network. Each PC in the MORSE network is represented by a different MORSE address. Other applications may also be connected via the LAN network in this way.

Configuration:

MR25ET has for example the address 690F5600,
The PCs use addresses 690F5701, 690F5702, 690F5703, ...

In the Ne menu we set the wide table, which will route packets for the PC to the Link output. Switch on address multiplication, which sends packets originally routed to Link to the User output, and connect the User and Service output to E00:

```
>>
  Nodes:
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 004C9D8F    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) 690F5600 L  E00 E00| -  R01|0 1 0 0| 15 SERV  OFF 304  30
(2) 00000000    S01 S02| -  R02|0 0 0 0| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000     -  S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30
```

The routing wide table sends packets which have the PC destination address to the actual address of the node:

```
>>
Wide retab. No 1
57to:5600
```

The EIe menu remain in the default condition:

```
>>
 Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1   NO AR          ON  OFF NONE
```

If we are working in the LAN network, in contrast to case (4)(A) the selection of the IP addresses of the application (PC with Setr) must also suit this network, for example:

```
  IP address - 192.168.0.119
  mask     - 255.255.255.0
```

In the Epe menu we select the IP address in accordance with the LAN network:

```
>>
Internet Protocol:
(i)p adr:C0A80044h     ... i.e. 192.168.0.68
(g)ateway adr:00000000h
(n)et mask:FFFFFF00h
(G):0000
(d)sc: 6039
AR(P) parameters
parame(t)ers
de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>t
```

(5)(B) By selecting (s)Art in the parameters we activate MAS, (B)ase and (M)ask determine the area in which MORSE addresses of connected Setr´s can be selected:

```
  INTERNET PROTOCOL GATEWAY:
M-IP-M:
  (A)rt:0; write (E)nable:ON
  (r)epeats:0000  (t)imeout:0  (p)roxy timeout:0s
  (f)rag size:400bytes  (g)lue (append) up to:0packets
```

```
IP-M-IP:
  (I)PArt:0
  (m)ask:00000000  IP(F)rag. size:552
MAS:
  (s)Art:1; write (e)nable:ON
  (B)ase:690F5700  (M)ask:000000FF
     BEWARE! IF YOU CHANGE CONTENT OF THE ART TABLE,
     YOU SHOULD RESTART (INIT) THIS PROTOCOL!
(P)inger (S)tatus
Firewall (T)ype:0000 A(d)dress:00000000 Mas(k):00000000
(q)uit
```

Start setr in windows using the command:
setr -pIP192.168.0.68 -pm8000 -pw690F5701

where -pIP is the MR25ET address, -pm is the application port number, -pw is the MORSE address, under which the PC with Setr will be known in the MORSE network. Other Setr may also be started at the same time on other PCs, however it is necessary that they all use a different MORSE address. After starting Setr we can look at how the Art table was automatically filled, here 2 Setr were started:

```
>>
ART No 1:
items: 2
default gw: 00000000 (0.0.0.0        )
dest:    gw:
690F5704 1F4000EB (105.15.87.4     31.64.0.235   )
690F5701 1F400077 (105.15.87.1     31.64.0.119   )
```

Connection example.

Routing, which ensures the sending of packets for the PC (here 690F57xx) to the MR25ET address (here 690F5600), must be set in the destination CU. In the CU 690F5601 it is table wide:

```
690F5601>
Wide retab. No 1
57to:5600


690F5601h>!

690F5601h>

u E00   690F5600    R01
30/ 70  690F5601   serd

 serd   690F5601    R03
29/ 16  690F5600  u E00
690F5601h>

Monitoring: Sequence error detected. Got 1 instead of 4.
09:12:57.045|690F5601 690F5600|690F5601 690F5701|0EE  RFTX     8*99 2dat
09:12:57.092|690F5600 690F5601|00000000 1FD80031|0EE*27~ 17    0|06  ack
09:12:57.171|690F5600 690F5601|690F5701 690F5601|00E*29* 16   24|9B 2dat
09:12:57.172|690F5601 690F5600|690F5601 690F5701|00E  RFTX     0*06  ack
```

(6) Packet transfer to the MORSE network

Set up routing and test the connection with a remote CU.

```
  690F5601h>!

u E00   690F5600    R01
31/ 77  690F5601   serd

 serd   690F5601    R01
30/ 74  690F5600  u E00
690F5601h>

690F5601>sts
```

In the remote CU we can set the async link protocol and direct it to the address of our application. (Caution, should the addresses differ here by more than the two lower bytes, then we have to use the SIe menu and translation by means of the Art tables.) In the case of (4)(A) the address of the application is 690f5600, in the case of (4)(B) it is 690F5701 or 690F5704 ...
Start monitoring of the port with async link in the remote CU, route it to the address of our application.

Test the connection from Windows to the remote async link:

```
690F5601h>mtu
Send type :09
690F5601h>
09:26:30.670 tx      4 | S00
6161 6161
```

On monitoring we see that Setr, using command "mtu", sent a message to MORSE address 690F5601 which carries four ASCII characters "aaaa".

Test the connection from the remote link to the application:

```
690f5601>in                   ... use the function Channel data send
690F5601>ah0123456789ABCDEF   ... insert some data (here hex)
690F5601>s                    ... start

690F5601h>O.K.
690F5601h>SETR: Packet type 09. 690F5600 690F5601  0C89    8
0123 4567 89AB CDEF           ... application Setr received and printed the data

09:39:25.359 rxsim   8 | S00
0123 4567 89AB CDEF           ... monitoring input of data to the remote CU
```

Note:
This is the connection with the async. link in the remote CU. There may be a lot of CUs of MORSE networks connected in this way and each can use a different protocol as necessary. Their data is then processed by the central application AAA connected through MAS, which distinguishes protocols best according to the respective MORSE addresses. This system integration is thus already contained in the MORSE system.

If we want to view the format of the UDP packet we can start Setr with the parameter -mh and Setr will print data in the UDP datagrams. It is a good idea to switch off remote monitoring for this activity so that Setr does not monitor monitoring packets. On the contrary local monitoring in the CU, which contains MAS is switched on here. Again function "mtu", which generates 4 ASCII characters "aaaa" in the Setr application is used:

```
690F5601>mtu

Send type :09
tx 12 to:10.0.0.2, 8888
0000 8905 690F 5601 6161 6161
```

... application Setr is started with parameter -mh. It sends a UDP datagram containing the MORSE pseudoframe 0000 8905 690F 5601 6161 6161 with destination MORSE address and data 6161 6161 to IP address 10.0.0.2 and to port 8888.

```
21:55:46.613 rsi:0036 rx|0002A94CA177 |00105A925F8B | IP/UDP/MOR/USR/DATA
0002 A94C A177 0010 5A92 5F8B 0800 4500 0028 B900 0000 8011 6DC2 0A00 0001 0A00
0002 1F40 22B8 0014 9EF2 0000 8905 690F 5601 6161 6161
```

... monitoring of the datagram entering the Ethernet channel contains, amongst others:

```
source IP address  0A000001
destination IP address    0A000002
source port        1F40 hex = 8000 dec
destination port          22B8 hex = 8888 dec
MORSE pseudoframe    0000 8905 690F 5601 6161 6161

RF mon      |toa      frm     |dst      src     |lNo!DQ!RSS size|TT N
21:55:46.617|690F5601 690F5600|690F5601 690F5600|013  RFTX     4*89 5dat
6161 6161
```

... monitoring of data sent through the MORSE radio channel

Monitoring through MAS.

When routing monitoring from the remote CU to the Setr connected through MAS it is necessary to insert the MORSE address of the PC with Setr into the "ise" menu. In the above example of the single application (A) it was address 690F5600, in the case of (B) it is, for example, address 690F5701. Use option "L" to insert it into the remote CU:

```
690F5601>
ise 0N1
    0aL     (type 'L' for local address)
```

The resulting menu "ise" contains the PC destination address (MORSE):

```
690F5601>
System channels:
(Service 'iMo' works for s0 and s1 only)

id|--Node--addr------timeout---size---s(e)c--
(0)  1    690F5701     888     400     ON
(1)  0    004C9D8F     888     400     ON
(2)  0    004C9D8F     888     400     ON
(3)  0    004C9D8F     888     400     ON
(4)  0    004C9D8F     888     400     ON
(5)  0    004C9D8F     888     400     ON
```

We then route monitoring to system channel "0".

(7) Using Setr switch off remote and local monitoring, switch off Setr, start application AAA and now only observe how everything operates. The application runs on UDP port 8000 and communicates against UDP port 8888, which is used by the CU. The format of data is described in another document (Format of UDP datagram IPGW for MORSE).

Brief summary of the steps described above for connecting the application through the MORSE Application Server (MAS):

(1) IP address in PC:
IP address - 10.0.0.1
mask - 255.255.255.0

IP address in CU:
Epe i0A000002 ... i.e. 10.0.0.2
nFFFFFF00 ... i.e. 255.255.255.0

(2)
ping 10.0.0.2

(3)
setr -pIP10.0.0.2

ping from CU to PC:
Epe tP t0A000001 s e p

(4)(A) Connection through MAS with one address:

```
Ne 1a 690F5600
   1uE0
   1sE0
EIe f
```

(5)(A) Setting MAS:

```
Epet t s1
       B690F5600
       M00000000
```

setr -pIP10.0.0.2 -pm8000 -pw690F5600

Content of Art:

```
dest:    gw:
690F5600 1F400001 (105.15.86.0     31.64.0.1 )
```

(4)(B) More addresses, connection of Setr through LAN:

```
Ne 1a690F5600
   1MLn
   1uE0
   1sE0
   1w1
Tw1
57to:5600
EIef
```

(5)(B) Setting MAS:

```
Epe iC0A80044
    nFFFFFF00
    t
     s1
     B690F5700
     M000000FF
```

setr -pIP192.168.0.68 -pm8000 -pw690F5701

Content of Art:

```
dest:    gw:
690F5704 1F4000EB (105.15.87.4     31.64.0.235 )
```

```
  690F5701 1F400077 (105.15.87.1     31.64.0.119 )

  Example of connection:

  destination CU 690F5601:
    Tw1
    57to:5600

    u E00   690F5600    R01
    30/ 70  690F5601   serd

     serd   690F5601    R03
    29/ 16  690F5600  u E00
```

(6) Test connection from Setr through MAS to the remote async link:

```
  690F5601h>mtu
  Send type :09
  690F5601h>
  09:26:30.670 tx      4 | S00
  6161 6161
```

Format of UDP datagram IPGW for Morse.

This datagram is used above all for Morse Application Server (MAS) and
for internal use in RACOM for IP Retranslation of Morse packets (M-IP-M).
IPGW = Internet Protocol Gate Way.

The UDP datagram wraped in the IP-datagram has this structure:

| IP header | UDP header | MORSE pseudoframe |

IP and UDP header are described in the literature, they are mentioned here
only for better orientation. The main part MAS follows this paragraph.
::::::::::::::::

.....
IP header (including the example):
.....

| IP header |
|vers/16|IPhlen/16| No/16 |frag/16|live/16|hchs/16| src IP/32 | dst IP/32 |
4500 0060 0014 0000 4011 D3DF C0A8 2105 C0A8 4444

Meaning of items of IP header:

vers/16	ver/4	- version of IP protocol (now 4)
	headlen/4	- length of IP header (here 5 words by 32 bits)
	type serv/8	- type of desired service for datagram transmission
IPhlen/16	number of bytes in all IPGW frame, hex (here 60 hex = 96 dec)	
No/16	order number, datagram identification, hex	
frag/16	fragmentation:	
	res/1	reserve
	no/1	1=datagram fragmentation forbidden
	next/1	1=next fragments follows
	offset/13	fragment offset- position of beginning of fragment data part in regard of underived datagram(in bytes)
live/16	time to live/8	- lifetime of datagram in seconds, by passing through router it is decremented at least by 1, at =0 it is discarded
	protocol/8	- determines the protokol of higher level, which report is contended in data part of datagram
hchs/16	checksum including IP head only	
src IP/32	src IP - source address,	see menu Epe, (i)p adr
dst IP/32	dst IP - destination address,	see menu Epe, (i)p adr

.....
UDP header:
.....

| UDP header |
| src port/16 | dst port/16 | UDPlen/16 | res/16 |
22B8 22B8 004C 0000

Meaning of items of UDP header:

src port/16	source port UDP (22B8 hex = 8888 dec)
dst port/16	destination port UDP
UDPlen/16	length of UDP packet in bytes(UDP header + pseudoframe),hex (here 4C hex = 76 dec)
res/16	reserve

.....
MORSE pseudoframe:
.....

According to first bite of pseudoframe is IPGW frame divided into 2 groups:
A) first bite is zero - Morse Application Server
B) first bite is one - proprietary for Racom, UDP frames for IP
retranslation of Morse traffic i.e. for packet route direction:
from MORSE through IP to MORSE (M-IP-M)

::
A) MAppS - Morse Application Server
::

| MORSE pseudoframe |
|flags/16|PT/8|D/1| R/4 | No/3| addr/32 | data |
| 0000 | 09 | 0 | 0000 | 000 |74490101 | 4500 003C ... |

```
Meaning of items:

flags/16  T/4 - 0x0 = 0000 - UDP datagram type MAppS
          R/4 - 0x0        - reserve
          U/8 - subtype - 0x00 - user data
                          0x01 - seek/delete format, for
                                 internal use in Racom only
PT/8             Morse packet type     |
D/1              DTE bit 0- DCE sending|
                         1- DTE sending| this part is coincidently
R/4              reserve (0000)        | similar like net level packet
No/3             packet order number   | of MARS-A protocol
addr/32          source address        |
data/(UDPlen-0x10)  data transmitted   |
     [   byte    ]

Used example:
>>
Monitoring: source 74490102|6.
14:52:05.583|eth:TX  110 |000000 000000|0002A9 5841E9| IP/UDP/MOR/USR/???
4500 0060 0014 0000 4011 D3DF C0A8 2105 C0A8 0444 22B8 22B8 004C 0000
0000 0900 7449 0101 4500 003C F900 0000 2001 9B45 C0A8 6420 C0A8 210A
0800 855B 0200 C600 6162 6364 6566 6768 696A 6B6C 6D6E 6F70 7172 7374
7576 ...
```

APPENDIX:

MORSE pseudoframe B), for internal use only. These datagrams are used for M-IP-M communication i.e. from MORSE network through IP network and to the MORSE again.

B1) UDP data frames and fragments - 0xD,0xB,0xF,0xE
B2) UDP appended frame - 0xA
B3) UDP control frame - 0xC

```
.........................................
 B1)     UDP data frames and fragments
.........................................

|                      MORSE pseudoframe
| flags/16 | No/16 | to/32   |  from/32 |type/16| dest/32 |  src/32 |
    D200     0025   690F8909   690F8101   0C89   690F8909   690F8101

       MORSE pseudoframe           |
| [mob/32] |    data      | crc/16 |
  ........      AAAA         8964

Meaning of items:
flags/16  T/4 - 0xD  data frame
              - 0xB  fragments begin
              - 0xF  fragment
              - 0xE  fragments end
          A/1  appended frame
          r/1  repeated bit (1 if the packet is repeated)
          S/1  security bit (1 requests ACK)
          P/1  transmitter problem bit
          R/4     reserve
          pver/4  protocol version
No/16        Link No
to/32        to   address
from/32      from address
type/16      packet full type
dest/32      destin. address
src/32       source  address
mob/32       mobile  address, for mobile service only,
                             i.e. if (type&0xE000) is nonzero
data/(UDPlen-0x20) [byte]    data transmitted
crc/16                    Cyclic Redundancy Check




.........................................
 B2)      UDP appended frame
.........................................
```

```
|                    MORSE pseudoframe                    |
| flags/16 | res/5 | lenA/11 | No/16  |  data  | crc/16 |
    AA00          0004          0005     AAAA     ....

Meaning of items:
flags/16  T/4 - 0xA  appended frame
          A/1  appended frame
          r/1  repeated bit (1 if the packet is repeated)
          S/1  security bit (1 requests ACK)
          P/1  transmitter problem bit
          R/4     reserve
          pver/4  protocol version
res/5         reserve
lenA/11       length (No + data), byte
No/16         Link No
data/(UDPlen-0x10) [byte]    data transmitted
crc/16                       Cyclic Redundancy Check


.............................................
 B3)      UDP control frame
.............................................

|               MORSE pseudoframe                       |
| flags/16 | CN/16 | Cf/16 | frm/32 | toa/32 | crc/16 |
   C100      0025    D200   690F8101 690F8909   EAAB

Meaning of items:
flags/16  T/4 - 0xC  control frame
          A/1  appended frame (=0)
          R/1  reserved
          CT/2 - control type
               - 00 NONE
               - 01 ACK
               - 10 REJ
               - 11 NAK
          R/4     reserved
          pver/4  protocol version
CN/16       control link No (link No of confirmed frame)
Cf/16       control flags   (flags of confirmed frame)
frm/32      from address
toa/32      to address
crc/16      Cyclic Redundancy Check


Used examples for appendix (B1,B3):
>>
12:12:16.768|eth:TX 0068 dst:0002A95841E9|src:0002A9588C21|
IP/UDP/MORSE/RET/DAT
4500 0036 0028 0000 4011 F934 C0A8 0001 C0A8 0009 22B8 22B8 0022 0000 D200
0025 690F 8909 690F 8101 0C89 690F 8909 690F 8101 AAAA 8964
12:12:16.773|eth:RX 0060 dst:0002A9588C21|src:0002A95841E9|
IP/UDP/MORSE/RET/CTL/ACK
4500 002C 0028 0000 4011 F93E C0A8 0009 C0A8 0001 22B8 22B8 0018 0000 C100
0025 D200 690F 8101 690F 8909 EAAB 0000
```

PPP Protocol for MORSE.

1. Introduction

The PPP protocol serves for connecting 2 points of the IP network over the serial link in the IPGW format.

It provides similar working modes as the Ethernet channel in the MR25ET.

Also see the Ethernet article in the MORSE Firmware document, which is available at www.racom.cz, in chapter
Software / Service Terminal / MORSE Firmware-Documentation / Ethernet.

2. Data Format

PPP protocol data contains an IP datagram with a PPP flag:

ppf/8	lch/8	IP datagram	fcs/16	ppf/8
7E	21	.....	....	7E

where:

ppf	flag PPP
21	LCP header
IP datagram	(packed into the escape sequence) An informative description is given in the article: "Format of UDP datagram IPGW for Morse"
fcs	frame check sequence (checksum PPP)

Example of PPP protocol data:

```
|ppf|lch|                  IP header                        |   UDP header
 7E  21  4500 0036 000F 4000 4011 B814 C0A8 00A1 C0A8 00A2 22B8 22B8 0022

    |                   pseudoframe                                    |
FA55 D000 0000 690F 12A2 690F 12A1 0E09 690F 12A2 690F 12A1 AAAA C5D7

| fcs  | ppf |
  083C   7E
```

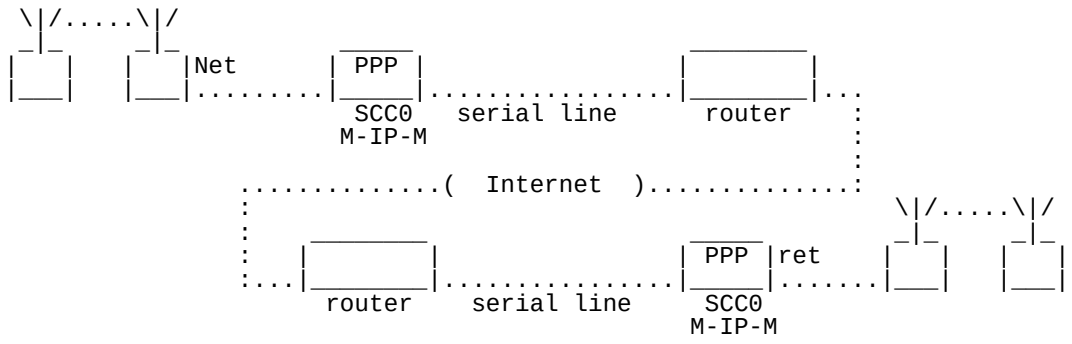
See - RFC-1661, RFC-1662
- Format of UDP datagram IPGW for MORSE

3. Implementation of the Protocol in the MORSE system

The PPP protocol enables communication in three modes:

3.1. M-IP-M, i.e. routing via IP

The section running along the IP network is inserted into the route of the packet in the MORSE network. It is used for example when checking a remote MORSE network.



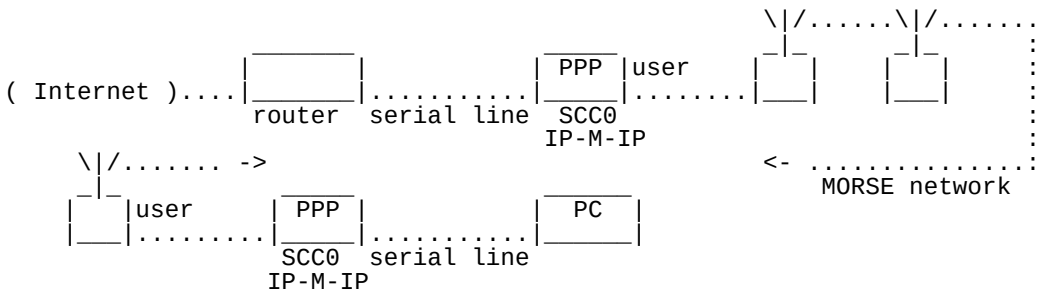
Configuration - the network output of the node (Net) goes to the SCC with the PPP protocol
- the routing output SCC (ret) goes to the node

A packet exiting a MORSE network node is given a "to" MORSE address and gets to the SCC channel via the node network output. Here it is assigned an IP address, found in the (A)rt table according to its "to" MORSE address. Then the packet is packed into a UDP datagram, PPP flags and a checksum are attached and the packet is sent along the serial link to the router.

In the destination router (IP host) a check is made as to whether the IP network route is also defined here in the opposite direction. If not the packet is rejected (REJ), if yes the UDP packing is removed from the received packet. An original MORSE packet obtained in this way goes via the SCC routing output to the MORSE node with the "to" address from where it continues through the MORSE network.

3.2. IP-M-IP, i.e. IP2MORSE

The section running along the IP network is inserted into the route of the packet in the MORSE network. It is used, for example, for access to the Internet via the MORSE network.



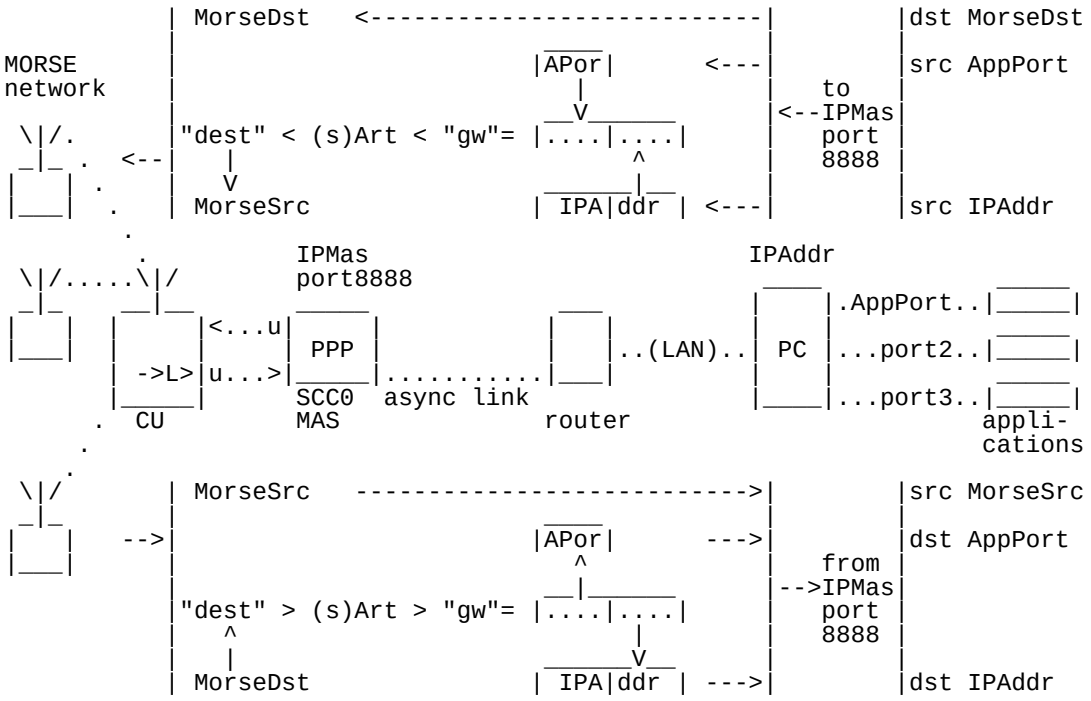
Configuration - user output of the node (user) goes to the SCC with PPP protocol
- user output SCC (user) goes to the node

The IP address of the packet coming from the IP network is found in the (I)Part table.
- If it is found the respective gw is used as the MORSE destination address.
- If it is not found and if it agrees with the actual IP address within the range of the mask (m)ask, then the MORSE dest address is made up of the actual MORSE address and the lower part of the IP address within the range of the (m)ask.
- In other cases the default gw from the (I)PArt table is used.

In the destination node of the MORSE route the packet is sent to the SCC channel through the user output. The MORSE pack is removed and the UDP packet is sent to the serial link.

3.3. Morse Application Server MAS, i.e. M-MAS-IP

MAS mediates communication between several applications in the connected LAN network and MORSE networks.



The HW route is represented in the diagram in the centre. The MORSE

network, from which packets arrive to the central CU with the PPP protocol and configured MAS, is on the left.
From here via the asynchronous link to the router, which is connected to the computer via the LAN (not however via WAN or the Internet). Various applications run on its ports and in this way communicate with MORSE network stations.

Processing of the addresses from the application to the MORSE network is indicated in the upper part of the diagram. The packet sent from the application carries the MorseDst address, which is used without change in the MORSE packet sent from the CU. Furthermore the packet is given a source IP address, IPAddr, and port number, AppPort. The packet with this set of information is sent to port 8888 of the MORSE Application server IPMas. Here the source address MorseSrc is created from AppPort and IPAddr as described below. The packet is given MorseDst and MorseSrc addresses and then sent to the MORSE network.

The destination MORSE station then uses this address MorseSrc for sending a reply to MAS, see the lower part of the screen. The MorseSrc address of the received packet passes through directly. dst AppPort and dst IPAddr, to which the reply is delivered, are created from the MorseDst address, see the next description.

The MAS function can also be approximated to operation of the RANEC monitoring system. Here the polled Morse network is connected via the CU in the MAS function with the LAN network, to which the RANEC server computer with six applications (demons) is connected. The demons successively call on the points of the Morse network via MAS, which assigns its source MORSE address to each packet leaving the demon to the network. Replies then arrive to these addresses from the network and using the "multiaddressing" (represented using ->L>) function all these replies are sent from the node via the user port to MAS. This then distributes them using the respective application around the LAN network according to these addresses, previously source and now dest.

Configuration - user output of the node goes to the SCC with PPP protocol
- user output goes to the node
- switch on the Multiaddressing function (Ne, 1MLo)

An IP packet sent from the application to a MORSE network point is given the IP address of the MAS server and port number 8888 dec. In MAS the application compiles a gw item from the source IP address and port number:
- port number sending application
  creates 16 MSB (Most Significant Bits) of the gw item
- 14 LSB (Least Significant Bits) of the IP address of application
  creates 14 LSB of gw item. These 14 bits form the address space of the LAN network, in which we can move.
- 15th bit is 0, in the case of a broadcasting packet it is 1
- 16th bit is reserved for the function of automatic assignment of addresses

According to gw a Morse address is found in (s)Art. This address is used in the assembled MORSE packet as the source address, the destination address is already contained in the incoming IP packet. A MORSE packet created in this way is sent via the SCC user output to the node and on to the MORSE network.

The packet from the MORSE network with a MORSE dest address of any of the applications arrives at the node from where it is sent, using the Multiaddressing function, via the user port to MAS. Here the gw item, from which the IP address and port number are created, is found in the (s)Art table according to the MORSE address of the application:
- 18 MSB from IPMas address supplemented by 14 LSB from gw create the IP address
- 16 MSB from gw define the application port number

The packet is sent over the LAN network to the application running on this IP address and port.

3.4. Examples of Communication

Example of the passage of a packet in the M-IP-M mode. AAAA data is sent from the node to the SCC, the PPP frame is then sent from the SCC:

```
>>
06:07:18.674 OUT    2 node (1): 690F110E 690F1109 |0809    0|
AAAA
06:07:18.677 tx   59 | S01
7E21 4500 0036 0000 4000 4011 B94F C0A8 0009 C0A8 000E 22B8 22B8 0022 E708
D000 0000 690F 110E 690F 1109 0809 690F 110E 690F 1109 AAAA E6A8 4885 7E
```

If communication does not take place a maintenance frame is sent every 20 seconds:

```
06:07:31.842 tx   14 | S01
```

```
7EC0 2109 0100 0800 0000 00C7 437E

06:07:52.340 tx   14 | S01
7EC0 210A 0100 0800 0000 0017 C97E
```

4. Setting Protocol Parameters

Arrangement of parameters in the Setr menu:

```
PPP parameters:
Wanted options:
(L)oc IP:C0A80009h 192.168.0.9
(R)mt IP:C0A8000Eh 192.168.0.14
Got options:
         C0A80009h 192.168.0.9
         C0A8000Eh 192.168.0.14
mr(u):1500
(h)ayes simulator:OFF
LCP m(a)gic:ON    obsolete LCP ma(G)ic (sv465..sv479):OFF
(F)lags:0
IPG(W)
(q)uit
>>W
```

```
(L)oc IP    - IP address of local end of the PPP link
(R)mt IP    - IP address of remote end of the PPP link
              Simply insert (L) and (R) into the protocol on one side of
              the link, the opposite protocol PPP obtains this data when
              making a connection with the LCP protocol.
              Similarly it is possible to insert on each side only (L)
              or only (R).
Got options - IP addresses got when making a connection with the LCP
              protocol. These addresses then appear in the transmitted
              packet.
mr(u)       - max. length of transferred packet after fragmentation,
              max.1600
(h)ayes simulator - emulates the function  of the telephone modem when
              the computer is communicating with the IP network through
              the MORSE channel
LCP m(a)gic - switches on the function of negotiating the magic number.
              Set using ON
obsolete LCP ma(G)ic - not used
(F)lags     - reserve, 0000
IPG(W)      - opens menu of parameters for IPGW. Parameters together with
              Ethernet channel parameters and their functions are
              described in the Setr description, i.e. in the
              "Morse Firmware Documentation" material.
              Available at www.racom.cz, chapter
              Software / Service Terminal / MORSE Firmware-Documentation.
```

Example of IPGW parameters:

```
  INTERNET PROTOCOL GATEWAY:
M-IP-M:
  (A)rt:1
  (r)epeats:0002  (t)imeout:500  (p)roxy timeout:0s
  (f)rag size:400bytes  (g)lue (append) up to:0packets
IP-M-IP:
  (I)PArt:0
  (m)ask:00000000  IP(F)rag. size:552
MAS:
  (s)Art:0
  (B)ase:00000000  (M)ask:00000000
     BEWARE! IF YOU CHANGE CONTENT OF THE ART TABLE,
     YOU SHOULD RESTART (INIT) THIS PROTOCOL!
  (P)inger (S)tatus
(q)uit
>>
```

Internet Protocol GateWay for Morse (IPGW)

Note: Here we do use "IP", or "internet" in sense ov IPv4. IPv6 support is under development.

There are three basic functions of IPGW:

1) MAS - Provides Morse Application Server, which can be used to run plenty of morse applications within LAN. Each application asks the morse server concurently for a morse address, or given application has its own fixed morse address preconfigured. This Application Server uses UDP datagrams and works with data format described below.

UDP port at the Application Server is usually port 8888 (recommended by Racom), UDP port at the application is arbitrary (i.e. port 8888).

There is no error correction mechanism in Application Server Protocol. That is why this protocol is suitable only for Local Area Networks. On the Internet is recomended more robust Morse retranslation on IP (see below).

2) IP-M-IP - Provides mapping of the IP address space to the Morse address space to achieve access to the internet from within the Morse network, including mobile IP applications. (I.e. browsing WWW pages using Morse network.)

Each IP datagram from Morse network does carry full IP address. IP datagrams from IP network are routed to Morse gateway found in AR table, or to address computed by mask mechanism, or to default gateway from AR table.

Single mobile Morse address can be a gateway for plenty of IP addresses, because IP datagrams in Morse network do carry both IP src and IP dst.

3) M-IP-M - Provides Morse retranslation on the Internet Protocol i.e. (retranslation through IP network for plenty of retranslation links - at least one hundred). Also supports retranslation links through IP proxy server (for the default gateway in the retranslation AR table only).

This protocol uses similair data format as the Application Server with field flags=0x8000. Leading bit is used to distinguish user and retranslation traffic.

Here - ad1), ad3) - we do not use TCP protocol, because of long timeouts of TCP. Morse network works with network round trip time in range 150ms..12000ms, which is less, than common TCP timeouts. TCP protocol also brings problems with packet synchronization, as it is connection orientated, and morse network is packet orientated (It is not easy to recognize packet start at TCP receiver after crash). TCP packet mode is rarely supported, although it is well defined.

IPGW is used in Morse network in Morse version of PPP protocol, and in Morse version of Eth/IP protocol.
It is planned for SLIP protocol for simple IP applications and also for UNIX Morse server (for big Morse applications).

bla, 20.2.2000.

How to configure Morse Application Server (MAS) :

Supposing PPP protocol active at port S0.

1) Turn off any ART stuff in PPP (Eth) protocol interface
in menu "SIe" - "0uAn" ("EIe" ...)

2) Configure MAS base address, mask and ART table
in menu "SPe" - "0tW","s3", "B690f0300","Mff"
(Ethernet version is similair.)

i.e.:

```
PPP parameters:
(L)oc IP:C0A80303 192.168.3.3
(R)mt IP:C0A80380 192.168.3.128
.
.
.
Morse App. Server:
(s)ART:3
(B)ase:690f0300 (M)ask:000000FF
```

This means that MAS will act as server for 256 morse addresses, in range
690f0300...690f03ff. Note that if MAS should serve more than one morse address,
you do need to use Multiadressing (menu "Ne", "1MLn"), and you have
to configure routing tables suitably.

In Art table No.3 should be information
how to translate incoming morse address to IP address and UDP port, i.e.:

```
ART No. 3:
items: 1
default gw:0
dest:    gw:
690f0301 0424004F
```

This means that a packet, coming from morse network to address 690f0301
will be routed to IP address 192.168.0.79 as UDP datagram to UDP port 1060.
Data format is described above.

How is this possible? IP address is computed as 17 Most Significant
Bits from local IP address and 15 Least Significant Bits from gw.
Note that 04fhex equals 79dec. 16th bit in gw is reserved for Racom's
internal usage and should be set to zero.

Udp port is 16 Most Significant Bits from gw. Note that 0424hex equals 1060dec.

Application at this IP address & UDP port should send morse datagrams
in correct data format to IP address 192.168.3.3 as UDP datagrams to UDP
port 8888(22B8hex) (This port number should be configurable, as future versions
of MAS will use more, than one port.).

You can test this using unix version of setr, for instance.
Type at unix prompt:

```
   setr -pIc0a80303 -pp8888 -pw690f0301 -pm1060
```

or more automatically using (yet) unpublished feature of setr -

```
   setr -pIc0a80303 -pw690f0301
```

or even more automatically

```
   setr -pIc0a80303
```

Setr does use service packet types (not user ones, as common application),
so you have to configure service output ("Ne" "1sS0") correctly.

For debugging purpose I do use often SCC channel monitoring and node
monitoring, all together "ime", "0h", "n0", "ndh". There you will see ip
datagram as data in SCC channel, and resulting morse packet. Note that
morse packets do transfer first dst, and then src, IP datagrams
do transmit first IPsrc and then IPdst.

26.10.2000, bla.

How to configure routing IP over Morse (IP-M-IP) :

Supposing PPP protocol active at port S0:

1) Turn off any ART address stuff for PPP (Eth) protocol
in menu "SIe" - "0uAn"

2) Configure IP2MORSE ART table & mask
in menu "SPe" - "0tW", "m0x000000ff","I1"
(Ethernet version is similair.)

3) Configure default gateway in ART table
"A" , "N1","w690f0102", "f"
or some explicit path
"A" , "N1","d0a010203", "w690f0103", "p"

I.e. If PPP protocol receives IP datagram from the serial line, it computes morse destination in folowing way:

First check the destination IP address against own IP address using mask.

1) If it fits, the morse address is computed from my own morse address and low part of IP destination address (using the mask).

Example: Supposing that your morse address is MyMorse=66112235, mask in PPP protocol parameters is mask=0xff, incoming frame from the serial line has IP dst IPdst=10.20.30.4, and Loc IP address in PPP protocol parameters (Got options) is MyIP=10.20.30.01. As (MyIP&~mask) equals (IPdst&~mask), then Morsedst=(MyMorse&~mask)|(IPdst&mask), i.e. Morsedst=66112204

2) If it does not fit, ART table (in your case No 1) is searched for specified IP path (i.e. 10.1.2.3). If it fits, the morse destination is the morse address from the gateway in ART (i.e.690f0103). If it does not fit any field in the ART table, the morse destination is default gateway from the ART, i.e. morse address 690f0102.

For debugging purpose I do use often SCC channel monitoring and node monitoring, all together "ime", "0h", "n0", "ndh". There you will see ip datagram as data in SCC channel, and resulting morse packet. Note that morse packets do transfer first dst, and the src, IP datagrams do transmit first IPsrc and then IPdst.

Please be careful to agree with your customers, how many packets of what length, how many TCP connections, or TCP files/streams/bytes they want to transfer per time unit. If they do plan to use other protocol, than TCP, make sure, that timeouts can be configured in large scale (up to 4-12 seconds).

26.10.2000, bla.

How to configure Morse retranslation over IP (M-IP-M) :

```
        This is very simple. This feature is pretty secure, if you do lock
radiomodems correctly. (To do this, you do need application
netlock - it is similair like setr.)

        Suppose, that you do have two radiomodems, one in Oslo, other in Lima.
Both are connected to the Internet using PPP protocol (or Ethernet):

        MR25 in Oslo with ip address 112.113.114.115 (0x70717273),
        morse address F1F2F3F4

        MR25 in Lima with ip address 240.241.242.243 (0xE0E1E2E3),
        morse address D1D2D3D4

        One of theese can be behind proxy IP server (IP masquerading). In this
case you will have to use proxy timeout (30sec) in PPP (Ehhetnet) protocol.
Opposite radiomodem must be on IP address visible from the Internet (i.e.
not behind proxy) You will have to configure firewalls correctly - to pass
through UDP datagrams for UDP port 8888.

        1)Go to Oslo.

Configure radiomodem's retranslation table.

   "Tg1"
   clear the table:
   "c"
   write path to lima for net D1XXXXXX:
   "pD1" "nD1D2D3D4"
   "r"
Global retab No. 1
D1 to D1D2D3D4
>>
   or default path to Lima:
   "pF1" "nD1D2D3D4"
   "r"
Global retab No. 1
F1 to D1D2D3D4
>>

Configure your own address.
   "Ne"
   "2aF1F2F3F4"
   retranslation table:
   "2g1"
   Net retranslation interface to serial channel S0 (with PPP protocol):
   "2NS1"
   "I"
   "w"
  Nodes:
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 0049AF63    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) F1000000    S00 S02| -  R01|0 0 0 1| 15 SERV  OFF 304  30
(2) F1F2F3F4    S01 S02| -  S01|0 0 0 1| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>

Configure PPP protocol.
   "SPe"
   "1t"
   "L70717273" (112.113.114.115)
   "R...something"
   "u1500"
   "W"
   "A1"
   "t1000"
   "r3"
   "q"
   "I"
   "w"

PPP parameters:
(P)ing
Wanted options:
(L)oc IP:70717273h 112.113.114.115
(R)mt IP:70717274h 112.113.114.116
```

```
Got options:
         00000000h 0.0.0.0
         00000000h 0.0.0.0
(h)ayes simulator:OFF
LCP m(a)gic:ON
mr(u):1500
(F)lags:0000

IP2MORSE:
(m)ask:00000000
BEWARE! IF YOU CHANGE THE ART TABLE, YOU SHOULD RESTART THE PROTOCOL!
(I)PArt:0
IP Retranslation:
(A)rt:1
(r)epeats:3 (t)imeout:1000
(p)roxy timeout:0s
(f)rag size:0bytes (g)lue (append) up to:0packets
Morse App. Server:
(s)Art:0
(B)ase:00000000 (M)ask:00000000
(q)uit
>>

Configure corresponding network interface.
   "SIe"
   "1rAn"
   "1uAn"
   "1rN2"
   "I"
   "w"
Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1 MASK 00000000/08 ON  OFF NONE
(1) 2   NO AR         | 2   NO AR          ON  OFF NONE
(2) 0   NO AR         | 1   NO AR          ON  OFF NONE

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>

Configure Address Resolution Table
   "A"
   "N1"
   "dD1D2D3D4"
   "wE0E1E2E3"
   "p"
   "r"

ART No 1:
items: 0
default gw: 00000000
dest:    gw:
D1D2D3D4 E0E1E2E3
>>
   Configuration of this Art can be done also automaticaly from Lima,
using proxy timeout in PPP protocol parameters in Lima. But if you
do have IP proxy server (IP masquerading) in Lima, you have to do this
using proxy timeout from Lima because in this Art table should be
IP address of proxy server and (invisible) UDP port dynamicaly
created by the proxy server.
So the record in Art table in Oslo will be dynamicaly updated acording
to state of IP proxy server in Lima.


        2) go to Lima, do similair things:

Global retab No. 1
F1 to F1F2F3F4
>>

   Alternatively default route:
Global retab No. 1
D1 to F1F2F3F4
>>

ART No 1:
items: 0
default gw: 00000000
dest:    gw:
F1F2F3F4 70717273
>>

  Nodes:
```

```
                                retab
Nid|address |M | u   s | L   N |l w n g|sTO Err  Cent vTO hTO
(0) 0049AF63    S02 S02| -  R00|0 0 0 0| 15 SERV  OFF 304  30
(1) D1D20001    S00 S02| -  R01|0 0 0 1| 15 SERV  OFF 304  30
(2) D1D2D3D4    S01 S02| -  S01|0 0 0 1| 15 SERV  OFF 304  30
(3) 00000000    S02 S02| -  R03|0 0 0 0| 15 SERV  OFF 304  30
(4) 00000000    S02 S02| -  R04|0 0 0 0| 15 SERV  OFF 304  30

de(f)ault (r)ead (w)rite
(I)nit (S)ync
(q)uit
>>

PPP parameters:
(P)ing
Wanted options:
(L)oc IP:E0E1E2E3h 240.241.242.243
(R)mt IP:E0E1E2E4h 240.241.242.244
Got options:
         00000000h 0.0.0.0
         00000000h 0.0.0.0
(h)ayes simulator:OFF
LCP m(a)gic:ON
mr(u):1800
(F)lags:0000
IP2MORSE:
(m)ask:00000000
BEWARE! IF YOU CHANGE THE ART TABLE, YOU SHOULD RESTART THE PROTOCOL!
(I)PArt:0
IP Retranslation:
(A)rt:1
(r)epeats:3 (t)imeout:1000
(p)roxy timeout:0s
(f)rag size:0bytes (g)lue (append) up to:0packets
Morse App. Server:
(s)Art:0
(B)ase:00000000 (M)ask:00000000
(q)uit
>>

Channel to Node Interface:
    retranslation     |   user                      lim
 id N  A t          m | N  A t Base     m  sec brc S   e
(0) 0   NO AR         | 1 MASK 00000000/08 ON  OFF NONE
(1) 2   NO AR         | 2   NO AR          ON  OFF NONE
(2) 0   NO AR         | 1   NO AR          ON  OFF NONE
```

   3)Test things from Lima
   "!hF1F2F3F4"
   "!"
   wait for the response

   4) If this works, configure morse routing in both MORSE networks (In Lima, and also in Oslo) properly using this link. Otherwise keep traveling from Oslo to Lima and back - and try to debug things.

        You do not need to spend energy for the journey to Oslo, and back to Lima, if you will use unix version of setr - it can be used via telnet easily.

        For debugging purpose I do use often SCC channel monitoring and node monitoring, all together "ime", "0h", "n0", "ndh". There you will see ip datagram as data in SCC channel, and resulting morse packet. Note that morse packets do transfer first dst, and the src, IP datagrams do transmit first IPsrc and then IPdst.

                1.11.2000, bla.

Simple Downloading using memcp.exe.

The macro mr25_cbl_le.bat contained in the sw package, for example mswin-i486-625-morse.zip, is suitable for saving module E from the computer to the radio (CU) using the service cable.

For it to function it is necessary for the directory, where the macro mr25_cbl_le.bat is located, to contain the file memcp.exe and the subdirectory fkl, containing the file me10.fkl. If the setr.exe program is running, switch it off.

Start the macro

```
mr25_cbl_le.bat
```

wait about 10 sec until a communication link is made and monitor the transfer of packets indicated by symbols ###. After transfer is complete switch off the download message using the Alt+F4 command and the module is ready for use in the CU.

Other macros containing mr25_cbl_... are used in the same way:

```
mr25_cbl_le.bat
mr25_cbl_ld.bat
mr25_cbl_lg.bat
mr25_cbl_lh.bat
mr25_cbl_la.bat
mr25_cbl_lb.bat
```

Below is a general description of the macro. For a more detailed definition see the article Downloading firmware to the CU - Introduction. Macros for downloading individual modules:

```
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\me10.fkl %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\md10.fkl %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\mg10.fkl %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\mh10.fkl %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\ma10.rfr %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as4 -pS -pb19200  -af.\fkl\mb10.fkl %1 %2 %3

memcp.exe          - calling the memcp program
-nt5000 -nr20      - timeout and repeat for transmission through the network
-ar -aE            - transferred modules are written directly into flash
-as8               - length of transmitted packets
-pS                - B-saver module active in the destination CU
-pb115200          - signalling rate over service cable
-af.\fkl\me10.fkl - calling source file me10.fkl
%1 %2 %3           - place for other possible parameters
```

When downloading various modules only the labelling of the source file me10.fkl changes. For increased security when transmitting module B the length of the packet is shortened to -as4 and the signalling rate is reduced to -pb19200. Similarly we can prepare the macro for the transmission of module mw10.rfr (caution, modules ma10.rfr and mw10.rfr have a different extension from the others):

```
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -pb115200 -af.\fkl\mw10.rfr %1 %2 %3
```

Downloading firmware into the CU - Introduction.

The CU is a common name for the Communication Unit of the MORSE system, which can be the MR25, MR25ET, MCM302, MR900, sw MORCE. The firmware is that part of the software, which is loaded into the CU. The second sw part are applications, e.g. SETR.exe used for the service communication between the PC and CU. The CU contains three memory spaces used for firmware modules:

1) B - Basic module
2) A or W - Air or Wire module
3) C - Loader
4) E or others - Main module, can exist in versions E, D, G, H, I...

ad 1) B (B-saver) is the basic module which comes up after resetting the CU, later the program goes into module E. While downloading the basic module be very careful, because in case of failure when downloading B, it is necessary to send the CU to RACOM for reloading. Often it is not necessary to download the new B module, but always check the sw version on the radio modem prior to downloading the software to the real radio network.

ad 2) A module is used in the case of downloading the main module (E) through the radio channel. The A module can be replaced by module W for downloading the main module over the wire link between the SCC of two CU.

ad 3) C (c-loader) is used in the MR25 for downloading via D-RAM.

ad 4) E module is the main program module which can be replaced by module D or G, H, I. These modules basically differ in the set of user protocols. The main module allows the transfer of the A or W or B module.

The firmware versions are numbered as they are developed, e.g. version 6.25 was created in May 2003. Using the command (s)ervice (v)ersion in Setr we obtain a list of present modules in CU e.g.:

```
  D625
  A602
  B625
  C625
```

It is not necessary to have equal numbers in all modules, but for the possible use of older modules you should consult with the sw section in Racom. The latest version of the A or W module is now (7/2003) the A602 or W602.

There are two ways how to load the firmware module into the CU:
A) directly into Flash memory
B) via D-RAM memory

ad A) The transferred data is written directly into Flash memory. The CU runs in another module, than the one being downloaded to. E.g. the CU runs in the A module and data is downloaded to the E module. During downloading (several minutes) the CU can perform store and forward relaying but not its other functions.

ad B) This way can only be used in those CUs which contain a lot of D-RAM memory, like the MR25ET, MCM302, MR900. The module E with the help of the C module is able to download to itself. The data is first written into D-RAM, Flash remains unchanged, and the CU can normally communicate with other members of the network. Only when the transfer is completed and data checked is the module rewritten, over a few seconds, from D-RAM into Flash. This way is safer than the previous one from the interference point of view.

Two kinds of software from the MORSE package can be used for downloading:
- the MEMCP program for downloading from the PC into the CU via the service cable, Ethernet or the MORSE network; see the article Downloading using memcp.
- the SETR program and its service "itl" for downloading from the CU into the second CU via the MORSE network; see the article Downloading using itl.

packets

Downloading Firmware using Memcp.

(version 616 and later)

The loading of sw into the CU occurs in 4 situations:

- from PC to CU (memcp + B-saver)
- from PC to CU containing D-RAM (MR25ET, MCM302, MR900) (memcp + C-loader)
- from CU to CU (itl + A/W)
- from CU to CU containing D-RAM (MR25ET, MCM302, MR900) (itl + C-loader)

The new module "loader" was created as a supplement to the modules B,A,W,E,D,G,H... The loader should always be loaded before being used. The module B-saver is rewritten from version 616. The old module B must be replaced by the new one.
The recommended arrangement of directories to meet the batch file from version 621:

```
  morse
   fkl
      ma10.rfr
      mb10.fkl
      mc10.fkl
      me10.fkl
      ...
      mw10.rfr
    cygwin1.dll
    memcp  .exe
    setr   .exe
    mr25_cbl_le.bat
    ...
```

The files .fkl are used for downloading. Only the modules ma10, mw10 are in the version .rfr.
When working in Linux, use "a_memcp" instead of "memcp".

Downloading from the PC to the CU (memcp + B-saver)

- source = PC
- destination = any CU
- locally via the service cable (also via Ethernet or through the MORSE network for downloading to B, A, W module)

The module E runs when downloading to B, A, W. This is why both versions, local or remote, can be used with the -da parameter:

```
    memcp -af.\fkl\mb10.fkl -as8 -ar -aE
    memcp -af.\fkl\mb10.fkl -as4 -ar -aE -da690F5502
```

The parameter -as4 causes the creation of short packets in air (4x128byte).
The parameters -ar -aE cause writing into flash memory directly.
Using this command the module B-saver is updated. For this step it is advisable to disconnect other cables (Ethernet) to reduce the danger of interference.

Now we should use the B-saver for writing all modules (D, E, G, H, B, A, W):

```
    memcp -af.\fkl\me10.fkl -as8 -pb115200 -ar -aE -pS
```

The parameter -pS switches the CU to the module B-saver. This mode can only work locally, so the parameter -da cannot be used.
The parameter -pb115200 switches to the high communication speed. For loading of the module B-saver we omit this parameter and so the safer default speed of 19200 bit/s will be used.

For downloading ma10.rfr or mw10.rfr the same possibility can be used, like for downloading mb10.fkl. Using -pS we work locally with the B-saver or by omission of -pS we run the main module and remote access can be chosen:

```
    memcp -af.\rfr\ma10.rfr -as8 -pb115200 -ar -aE -pS                or
    memcp -af.\rfr\mw10.rfr -as8 -pb115200 -ar -aE     -da690F5502
```

Batch files:

For downloading, a set of batch files is prepared in the sw pack e.g. mswin-i486-625-morse.zip. The names and contents of files designated for work with the B-saver in the MR25 follow:

```
   mr25_cbl_le.bat
   mr25_cbl_ld.bat
   mr25_cbl_lg.bat
   mr25_cbl_lh.bat
```

```
   mr25_cbl_lb.bat
   mr25_cbl_lb_up.bat

memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -af.\fkl\me10.fkl -pb115200 %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -af.\fkl\md10.fkl -pb115200 %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -af.\fkl\mg10.fkl -pb115200 %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -af.\fkl\mh10.fkl -pb115200 %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8 -pS -af.\fkl\ma10.rfr -pb115200 %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as4 -pS -af.\fkl\mb10.fkl -pb19200  %1 %2 %3
memcp.exe -nt5000 -nr20 -ar -aE -as8     -af.\fkl\mb10.fkl -pb19200  %1 %2 %3
```

We can see, that the batches contain the -ar -aE parameter for directly writing into flash memory and the -pS parameter, which activates the B module. Downloading is done over the service cable. The -pS is omitted in the last batch, so the E module will be used (if the existing B module is older than sw616).

```
   rr_la_net25.bat  +  -pb115200
   rr_lb_net25.bat  +  -pIP192.168.0.1

memcp.exe -nt5000 -nr20 -ar -aE -as8   -af.\fkl\ma10.rfr + -pb115200        %1
%2
memcp.exe -nt5000 -nr20 -ar -aE -as8   -af.\fkl\mb10.fkl + -pIP192.168.0.1  %1
%2
```

The module E is used for downloading in these cases (-pS is missing) and so the connection via the service cable or via Ethernet is optional. After bringing the filename to the command line (Ctrl + Enter) we should attach either the -pb115200 parameter for downloading via the service cable or the -pIP192.168.x.x parameter for downloading via Ethernet. The hexadecimal form -pICOA80101 can be used as well. The E module allows the use of parameter -da690F0102, which defines the destination address for downloading in the MORSE network.

Downloading from the PC to the CU containing D-RAM (memcp + loader)

- source = PC
- destination = CU with D-RAM (MR25ET, MCM302, MR900)
- modules C, D, E, G, H,... but no B, A, W

Downloading is done in two steps. First the short C-loader is loaded and then with help of this loader the main module is downloaded.

- locally via service cable:

```
    memcp -af.\fkl\mc10.fkl -as8 -pb115200
    memcp -af.\fkl\me10.fkl -as8 -pb115200
```

- through local CU and the MORSE network in 690F5502:

```
    memcp -af.\fkl\mc10.fkl -as8 -pb115200 -da690F5502
    memcp -af.\fkl\me10.fkl -as8 -pb115200 -da690F5502
```

- by Ethernet connection:

```
    memcp -af.\fkl\mc10.fkl -as8 -pIP192.168.0.68
    memcp -af.\fkl\me10.fkl -as8 -pIP192.168.0.68
```

- through LAN and MR25ET and MORSE network to 690F5503:

```
    memcp -af.\fkl\mc10.fkl -as8 -pIP192.168.0.68 -da690F5503
    memcp -af.\fkl\me10.fkl -as8 -pIP192.168.0.68 -da690F5503
```

The transfer of the loader takes about 2sec, transfer of me10 takes about 45sec via the service cable or via Ethernet about 6sec. This is followed by the copying of transferred data from RAM to Flash memory (6sec) and finally it is reset.

Batch files:

```
   mr25_net_le_et.bat  +  -pIP192.168.0.1

memcp.exe -nt5000 -nr20   -as8   -af.\fkl\mc10.fkl + -pIP192.168.0.1
memcp.exe -nt5000 -nr20   -as8   -af.\fkl\me10.fkl + -pIP192.168.0.1
```

We should again attach the parameter -pb or -pIP, or alternatively -da. In the first step this batch writes the C-module into D-RAM memory (-ar -aE is missing), then copies it into flash and in the second step it does the same with the E module.

Transfer of the module from CU to CU (itl + module A/W)

- source any CU

- destination   any CU
- through MORSE network:
The work with itl is described in the article Downloading using itl.
Only a short overview is given here:

```
Setr  itl
      N1 d690F5502 x4
      mE    ... module transfered
      si    ... start test
      Aie   ... switches in module A
      fe    ... download
      Se    ... checksum
      Bie   ... switches via B into E
      p     ... stop test
```

Transfer of the module from CU to CU with D-RAM   (itl + C-loader)

- source   any CU
- destination   CU containing D-RAM (MR25ET, MCM302, MR900)
- the loader + module for transfer are prepared in the source CU
- through MORSE network:

```
Setr  itl
      N1 d690F5502 x4
      mm    ... module transfer
      yl    ... transfer of C-loader
      sie   ... start test
      fe    ... download
      p     ... stop test
      yk    ... transfer of kernel, e.g. me10
      sie   ... start test
      fe    ... download
      Ce    ... copy into flash
      p     ... stop test
```

The work with itl is described in Downloading using itl.

Comment 1: If new software (>sw600) is downloaded to the CU it is first NECESSARY to check the destination CU whether it contains the new (616 and later) module B ! If not, then download it, e.g. using the command:

```
    memcp -af.\fkl\mb10.fkl -ar -aE
```

The other modules can be loaded using this new B-saver.

Comment 2: Don't use sw600 to sw615 for loading old B < sw600 ! This can destroy sw in the CU. SW616 and later versions allow this operation. When new sw is replaced with old sw (< sw600), the configuration parameters will be lost.

Comment 3: The module ma10.rfr and mw10.rfr are contained in the set of sw602 firmware.

Downloading using itl.

The downloading of firmware over the radio channel is a relatively complicated task containing various dangers for the network. This is why it is only recommended for experienced operators.

The downloading mentioned here allows for the transfer of firmware modules between Communication units (CU) in the MORSE network. It uses the "itl" service of the SETR program, which allows both ways of loading the firmware module into the CU:

- directly into Flash memory, see article (1) and (2)
- via D-RAM memory, see article (3)

---

(1) Downloading of module A or W or B through module E.

These conditions must be accomplished:
- good connection between the source and destination CU, and a direct connection without retranslation is recommended
- the source CU contains this module (A or W), which should be transferred to the destination CU

Start the test memload using the command "itl" and fill in the next items:
(d):690F5513 - address of destination CU
(x):4   - number of 128 byte sectors transferred in one packet, for wrong conditions of the radio connection choose the lower number, e.g. x=2, for very good conditions use x=8
(m):A   - module transferred, A represents both A or W

```
  Memload:
  (N):1   (d):690F5513h
  (E)xternal flash:OFF
  ma(x) sectors:4   (m)odule:A   preset t(y)pe
  User module: fi(r)st:00300000   (l)ast:0037FF80
  (t)imeout:12000
  (s)tart r(e)port sto(p)
  go MORSE (A)/(W)
  (i)nit
  (f)ire (k)ill (c)ontinue
  check through (S)UM32
  go MORSE (B)
  (C)..modprobe with chksum
  (M)odprobe
  (q)uit
  >>
```

Continue with next commands:

(s) Enter - start test
(i) Enter - init in destination CU
(e) Enter - check the test report before work:

```
  This is Memload v1.03 response
  max. MF sectors per packet :4
  loading module: MORSE A
  target        : 690F5513h
  status :ready Check result: none
  begin  :  374000h
  end    :  37D800h
  current:  374000h
  talking to MORSE E
  timeout 12000
  Time elapsed: 0msec
  Transfer rate: nankbps
```

where:
loading module: MORSE A - the module (= A or W) transferred
talking to MORSE E   - the transfer will be done using module E (= E or D or other main module)

The transfer starts using (f)ire:
(f) Enter

We can follow the progress using:
(e) Enter

```
  ...
  status :waiting for memfill response Check result: none
  begin  :  374000h
  end    :  37D800h
  current:  378400h
  ...
```

If necessary, we can interrupt the process using (k)ill and go on using (c)ontinue.
When completed, the report looks like this:

```
  status :all sectors are transferred. Check result: none
  begin  :  374000h
  end    :  37D800h
  current:  37D800h
```

A checksum of the transferred module will be done using command:
(S) Enter
(e) Enter

The previous message appears, however containing the row:

```
  status :all sectors are transferred. Check result: O.K.
```

In the case, when the Check result is not O.K., stop the test and start again using the command (s)tart, see previous article.

Now stop the test in the source CU:
(p) Enter

Downloading of the chosen module is complete.

The summary of commands used follows:

```
  itl
  N1 d690F5513 x4
  mA      ... module transferred
  sie     ... test start
  fe      ... loading
  Se      ... check sum
  p       ... stop test
```

---

(2) Downloading of main module E (= E or D ...) or B through module A or W.

These conditions must be accomplished:
- good connection between the source and destination CU, and a direct connection without retranslation is recommended
- source CU contains this main module (E or D or ...), which should be transferred to the destination CU
- the destination CU contains module A, if we are going to transfer the main module on air, or
  the destination CU contains module W, if we are going to transfer the main module via wire link

Start the test memload using the command "itl" and fill in the next items:
(d):690F5513 - address of the destination CU
(x):4   - number of 128 byte sectors transferred in one packet, for wrong conditions choose the lower number
(m):E   - module transferred, E represents the current main module in the source CU (E, D, G, H, I)

Continue with the next commands:

(s) Enter - start test
(i) Enter - init in the destination CU
(A) Enter - switch the dest. CU in the module A
or (W) Enter - switch the dest. CU in the module W in case of downloading via wire
(i) Enter - init in the destination CU
(e) Enter - check the test report before work:

Sometimes it happens that a similar message appears:

```
  status :ready Check result: none
  begin  :  308000h
  end    :  371000h
  current:  308000h
                    E via E???
  talking to MORSE E
```

In this case switching from E to A module in the destination modem did not happen and we will repeat the commands:
(A) Enter - switch the dest. CU in the module A
(i) Enter - init in destination CU
(e) Enter - check the test report before work:

The right answer looks like this:

```
  status :ready Check result: none
  begin  :  308000h
```

```
  end    :  371000h
  current:  308000h
  talking to MORSE A(W)
```

where:
  talking to MORSE A(W)   - the transfer will be done using module A or W

The transfer starts using (f)ire:
(f) Enter

The transfer is completed, when we obtain after
(e) Enter
a reply containing:

```
  status :all sectors are transferred. Check result: none
```

A checksum of the transferred module will be done using command:
(S) Enter
(e) Enter

The destination CU is working in module A or W. In this state (e.g. after downloading has finished) it is possible to call this CU using
  ! Enter
but it is only able to execute the basic commands and the store and forward relaying. For example after a request "sv" the reply "A602" comes and after request:

(N)ode (e)dit Enter   the message:

```
  690F5513h> @  A/W: Service not available!     appears.
```

So after downloading has finished it is necessary to switch the destination CU in the main module again:
(B) Enter - restart of the destination CU is executed, the CU later
(i) Enter   goes into the main module (e.g. E module)
(e) Enter - checking if the destination CU is switched back into E module:

```
  This is Memload v1.03 response
  max. MF sectors per packet :4
  loading module: MORSE E
  target        : 690F5513h
  status :ready Check result: none
  begin  :  308000h
  end    :  371000h
  current:  308000h

                            E via E???
  talking to MORSE E
  timeout 12000
  Time elapsed: 0msec
  Transfer rate: nankbps
  >>
```

If this message appears, then the destination CU is O.K. and we can stop the test:

(p) Enter - stop the memload test in the source CU

Downloading of the main module is finished.

The summary of commands used follows:

```
  itl
  N1 d690F5513 x4
  mE      ... module transferred
  si      ... test start
  Aie     ... switch to module A
  fe      ... loading
  Se      ... check sum
  Bie     ... switch back to B and E
  p       ... stop test
```

When the module B is transferred it is important not to restart or switch off the CU or use the command "go MORSE (B)" until the transfer is successfully completed.

---

(3)   Downloading the main module E (= E or D ...) using the loader and kernel module.

These conditions must be accomplished:
- good connection
- the destination CU contains the D-RAM, so it is the MR25ET or MCM302 or MR900 (but not MR25)

- the source CU contains this main module (E or D or ...), which should be transfered in the destination CU
- the source CU contains the loader module mc10.fkl, best newly loaded using the command
    memcp -af./fkl/mc10.fkl -as8 -pb115200

Choose the test memload using the command "itl" and fill in the next items:
(d):690F5513 - address of the destination CU
(x):4   - number of sectors
(m):m   - module mode
(y):l   - module type loader, we can see that these addresses appear:
  User module: fi(r)st:02000002   (l)ast:02000002

(s)   Enter - start test itl
(i)   Enter - init in the destination CU
(e)   Enter - report, next message appears:
  status :ready; Check result: none
  begin : 37D200h
  end   : 37D684h
  current: 37D200h
  talking to MORSE E

(f)   Enter - start the transfer
(e)   Enter - after a few seconds the report looks like this:
  status :all sectors are transferred.; Check result: none
  begin : 37D200h
  end   : 37D684h
  current: 37D700h

(p)   Enter - stop test
(y):k Enter - switch to kernel (main module) transfer
(s)   Enter - start test itl once more
(i)   Enter - init in destination CU
(e)   Enter - report, next message appears:
  User module: fi(r)st:01000001   (l)ast:03000001

(f)   Enter - start the transfer of the kernel
(e)   Enter - after a few minutes the report looks like this:
  status :all sectors are transferred.; Check result: none
  begin : 308000h
  end   : 36A500h
  current: 36A500h

(C)   Enter - the transferred kernel in the destination CU is checked and copied from D-RAM to the Flash memory. It takes about 6 seconds.
(e)   Enter - report:
  status :pal O.K.; Check result: none
(p)   Enter - stop test itl

Downloading of the module is completed.

The summary of commands used follows:

```
  itl
  N1 d690F5513 x4
  mm      ... module transferred
  yl      ... loader module
  sie     ... test start
  fe      ... loading
  p       ... stop test
  yk      ... kernel module
  sie     ... test start
  fe      ... loading
  Ce      ... copy to flash
  p       ... stop test
```

Be careful when downloading the older main module (<602) instead of the new one (>=602). After this it is necessary to return the parameters to the default state (e.g. using "ca" and "cw" commands) and set the parameters again manually.

The versions A or W older than 602 in one CU do not work with new (602 and later) sw in the second CU. It is recommended to replace them by version A602 or W602 or later, if available.

The module B-saver is rewritten from version 616. The old module B must be replaced by the new one.

All operations can be done using remote access. In this way it is possible to download the firmware module step by step in the whole network.

When the main module is downloaded via module A, then the destination CU can

only fulfil basic functions, until it is rebooted again. The CU running in the A module can do store and forward relaying in the network, but in spite of this, it is recommendable to reduce the traffic in a heavily loaded network when downloading.

The (i)nit command in the "itl" can be used more often when we are in doubt about good progress of work, e.g. when switching to the A module.