```
// AM/FM Tuner an einen Syncronen Kanal binden
// 0x40, 0x00, 0x101, 0x2, 0x01, 0x01

unsigned char cmd[]={0x00,0x40,0x00,0x01,0x01,0x02,0x01,0x01};
CMessage newmsg;
newmsg.setSenderType(0x01);
const MostMessage* mostMsg = newmsg.getMostMessage();
memcpy((unsigned char*)mostMsg->data.bytes, cmd, sizeof(cmd));
CContext::getMainDispatcherContext().getNormalQueue().add(newmsg);
```

Oder so geht‘s auch:

```
MostMessage most;
CMessage newmsg(most);

most.senderType = 0x01;                                  // optical
most.data.rx_msg.FBlockID= 0x40;                         // Tuner
most.data.rx_msg.Instance= 0x00;                         // inst 0
most.data.rx_msg.FkID      = 0x101;                      // Allocate
most.data.rx_msg.OPType= 0x2;                            // StartResult
most.data.rx_msg.Length   = 0x1;
most.data.rx_msg.data[0]   = 0x1;
newmsg.setMostMessage(most);
MostMessage *mostMsg2=(MostMessage*) (newmsg.getMostMessage());
CContext::getMainDispatcherContext().getNormalQueue().add(newmsg);          // in die Q
```

```
MostMessage most2;
CMessage newmsg2(most2);

most2.senderType = 0x01;                                    // optical
most2.data.rx_msg.FBlockID          = 0x22;                 //Amplifier
most2.data.rx_msg.Instance          = 0x00;                 // Inst 0
most2.data.rx_msg.FkID              = 0x111;                // Connect
most2.data.rx_msg.OPType            = 0x2;                  // StartResult
most2.data.rx_msg.Length            = 0x0006;
most2.data.rx_msg.data[0]           = 0x01;
most2.data.rx_msg.data[1]           = 0x00;
most2.data.rx_msg.data[2]           = 0x00;
most2.data.rx_msg.data[3]           = 0x01;
most2.data.rx_msg.data[4]           = 0x02;
most2.data.rx_msg.data[5]           = 0x03;

newmsg2.setMostMessage(most2);
CContext::getMainDispatcherContext().getNormalQueue().add(newmsg2);
```

```
void CTunerController::tunerSetNextFrequency()
{
        MostMessage most;
        CMessage newmsg(most);

        most.senderType                 = 0x01;     // optical
        most.data.rx_msg.FBlockID       = 0x40;     // Tuner
        most.data.rx_msg.Instance       = 0x00;     // Inst
        most.data.rx_msg.FkID           = 0xc04;    // Allocate
        most.data.rx_msg.OPType         = 0x2;      // StartResult
        most.data.rx_msg.Length         = 0x0002;
        most.data.rx_msg.data[0]        = 0x01;
        most.data.rx_msg.data[1]        = 0x00;
```

Oder so geht‘s auch

```
void Startup()                  {SendMessage(0x101,MOST_OPTYPE_CMD_SETGET,0x01);}
void DemandStationInfo()  { SendMessage(0xc11,MOST_OPTYPE_CMD_GET, 0x01,0x00);}
void SetNextFrequency()   { SendMessage(0xc04,MOST_OPTYPE_CMD_SETGET, 0x01,0x00);}
void SetPrevFrequency()   { SendMessage(0xc04,MOST_OPTYPE_CMD_SETGET, 0x011,0x00);}
void SetFrequency(int kHz) {SendMessage(0x206,MOST_OPTYPE_CMD_SETGET, 0x011,(unsigned char)kHz);}
```

```
void CAmpController::setVolume(int value)
{
		volume = value;

// Soundsettings einspielen
// 0x022, 0x00, 0xc00, 0x2, 0x0b, 0x01, 0x00, 0x02, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
		MostMessage most3;
		CMessage newmsg3(most3);

		most3.senderType			= 0x01;		// optical
		most3.data.rx_msg.FBlockID		= 0x22;		//Amplifier
		most3.data.rx_msg.Instance		= 0x00;
		most3.data.rx_msg.FkID			= 0xc00;	// CompactSoundSettings (Mercedes Prop.)
		most3.data.rx_msg.OPType		= 0x2;		// StartResult
		most3.data.rx_msg.Length		= 0x000b;
		most3.data.rx_msg.data[0]		= 0x01;
		most3.data.rx_msg.data[1]		= 0x00;
		most3.data.rx_msg.data[2]		= 0x02;
		most3.data.rx_msg.data[3]		= (unsigned char) value;
		most3.data.rx_msg.data[4]		= 0x00;
		most3.data.rx_msg.data[5]		= 0x00;
		most3.data.rx_msg.data[6]		= 0x00;
		most3.data.rx_msg.data[7]		= 0x00;
		most3.data.rx_msg.data[8]		= 0x00;
		most3.data.rx_msg.data[9]		= 0x00;
		most3.data.rx_msg.data[10]		= 0x00;
		most3.data.rx_msg.data[11]		= 0x00;

		newmsg3.setMostMessage(most3);
		CContext::getMainDispatcherContext().getNormalQueue().add(newmsg3);
```

```
void CAmpController::getVolume()
{
		unsigned char cmd[]={0x00,0x22,0x00,0x0c,0x00,0x01,0x03,0x01,0x00,0x02};
		CMessage newmsg;
		const MostMessage* mostMsg = newmsg.getMostMessage();
		memcpy((unsigned char*)mostMsg->data.bytes, cmd, sizeof(cmd));
		CContext::getMainDispatcherContext().getNormalQueue().add(*mostMsg);
}

void CAmpController::channelMute(int channel)
{
		// Unmuten des Syncronen Kanal
		// 0x022, 0x00, 0x113, 0x2, 0x02, 0x01, 0x00
		MostMessage most4;
		CMessage newmsg4(most4);
```

```
		most4.senderType		= 0x01;		// optical
		most4.data.rx_msg.FBlockID	= 0x22;		// Amplifier
		most4.data.rx_msg.Instance	= 0x00;
		most4.data.rx_msg.FkID		= 0x113;	// Mute
		most4.data.rx_msg.OPType	= 0x2;		// StartResult
		most4.data.rx_msg.Length	= 0x0002;
		most4.data.rx_msg.data[0]	= channel;
		most4.data.rx_msg.data[1]	= 0x00;

		newmsg4.setMostMessage(most4);
		CContext::getMainDispatcherContext().getNormalQueue().add(newmsg4);
}
```

```
void CAmpController::setBass(int value)
{
		MostMessage most3;
		CMessage newmsg3(most3);

   most3.senderType			= 0x01;		// optical
   most3.data.rx_msg.FBlockID		= 0x22;		//Amplifier
   most3.data.rx_msg.Instance		= 0x00;
   most3.data.rx_msg.FkID		= 0xc00;	// CompactSoundSettings (Mercedes Prop.)
   most3.data.rx_msg.OPType		= 0x2;		// StartResult
   most3.data.rx_msg.Length		= 0x000b;
   most3.data.rx_msg.data[0]		= 0x01;
   most3.data.rx_msg.data[1]		= 0x00;
   most3.data.rx_msg.data[2]		= 0x04;					 // Mask
   most3.data.rx_msg.data[3]		= (unsigned char)volume;		// Lautstaerke!!!
   most3.data.rx_msg.data[4]		= (unsigned char)value;			// Bass
   most3.data.rx_msg.data[5]		= 0x00;					//Treble
   most3.data.rx_msg.data[6]		= 0x00;					//Balance
   most3.data.rx_msg.data[7]		= 0x00;		 //Fader
   most3.data.rx_msg.data[8]		= 0x00;
   most3.data.rx_msg.data[9]		= 0x00;
   most3.data.rx_msg.data[10]		= 0x00;
   most3.data.rx_msg.data[11]		= 0x00;

   newmsg3.setMostMessage(most3);
   CContext::getMainDispatcherContext().getNormalQueue().add(newmsg3);

}
```

```
void CAmpController::setTreble(int value)
{
    MostMessage most3;
    CMessage newmsg3(most3);


    most3.senderType                = 0x01;     // optical
    most3.data.rx_msg.FBlockID      = 0x22;     //Amplifier
    most3.data.rx_msg.Instance      = 0x00;
    most3.data.rx_msg.FkID          = 0xc00;    // CompactSoundSettings (Mercedes Prop.)
    most3.data.rx_msg.OPType        = 0x2;      // StartResult
    most3.data.rx_msg.Length        = 0x000b;
    most3.data.rx_msg.data[0]       = 0x01;
    most3.data.rx_msg.data[1]       = 0x00;
    most3.data.rx_msg.data[2]       = 0x08;     // Mask
    most3.data.rx_msg.data[3]       = (unsigned char)volume;        // Lautstaerke!!!
    most3.data.rx_msg.data[4]       = 0x00;     // Bass
    most3.data.rx_msg.data[5]       = (unsigned char)value;         //Treble
    most3.data.rx_msg.data[6]       = 0x00;     //Balance
    most3.data.rx_msg.data[7]       = 0x00;     //Fader
    most3.data.rx_msg.data[8]       = 0x00;
    most3.data.rx_msg.data[9]       = 0x00;
    most3.data.rx_msg.data[10]      = 0x00;
    most3.data.rx_msg.data[11]      = 0x00;

    newmsg3.setMostMessage(most3);
    CContext::getMainDispatcherContext().getNormalQueue().add(newmsg3);

}
```