

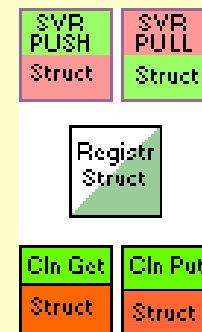
# user defined TINE structures in LabVIEW

trigger:

- we ask Phil to create a new data type: NAME128I

solution:

- make use of 'tagged' structure in buffered TINE server
- in and out data type in exports.csv (*struct.structurename*)
- register the structure on client and server side  
(using functions in tine.dll: *addFieldToStruct* and *sealTaggedStruct*)
- send/receive:  
    cln: *tputgetAnyArray*      srv: *push- / pullBufferedData*
- VIs ready for NAMExxxLong

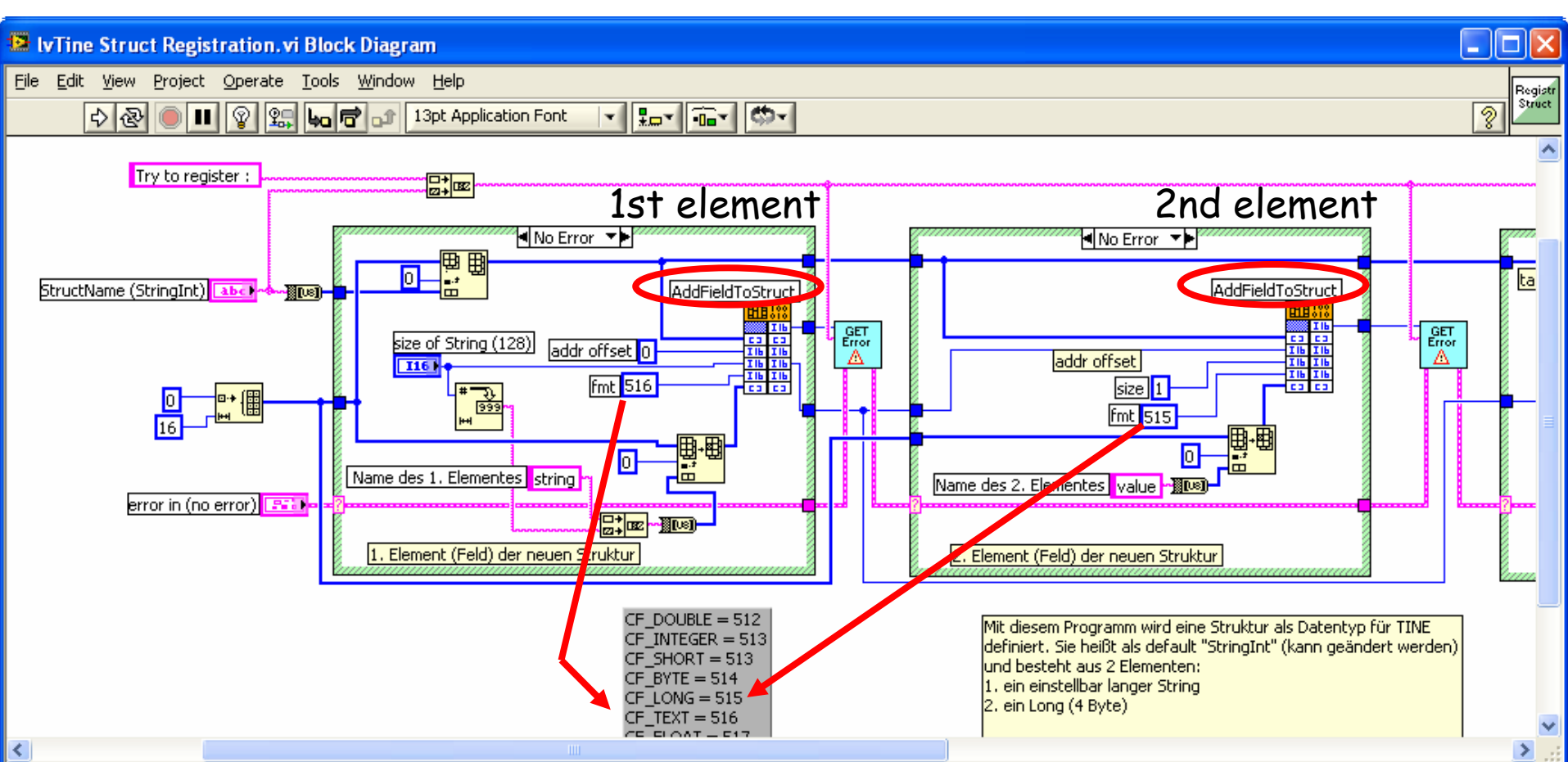


# How C did the registration:

```
#define STRING_SIZE 256
typedef struct
{
    char string[STRING_SIZE];
    int value;
} StringInt;

void registerStructs(void)
{
    short cc = 0;
    char errstr[256], tag[32];
    /* register the fields in a structure like this */
    strncpy(tag,"StringInt",TAG_NAME_SIZE);
    if ((cc=addFieldToStruct(tag,OFFSETIN(StringInt,string),STRING_SIZE,CF_TEXT,"string")) != 0) ccerr(cc);
    if ((cc=addFieldToStruct(tag,OFFSETIN(StringInt,value),1,CF_LONG,"value")) != 0) ccerr(cc);
    /* close the registration when finished */
    if ((cc=sealTaggedStruct(tag,sizeof(StringInt),100)) != 0) ccerr(cc);
    /* any others ? ...*/

err:
    if (cc != 0)
    {
        feclog("registerStructs : tag %s -> %s",tag,GetLastLinkError(cc,errstr));
    }
}
```

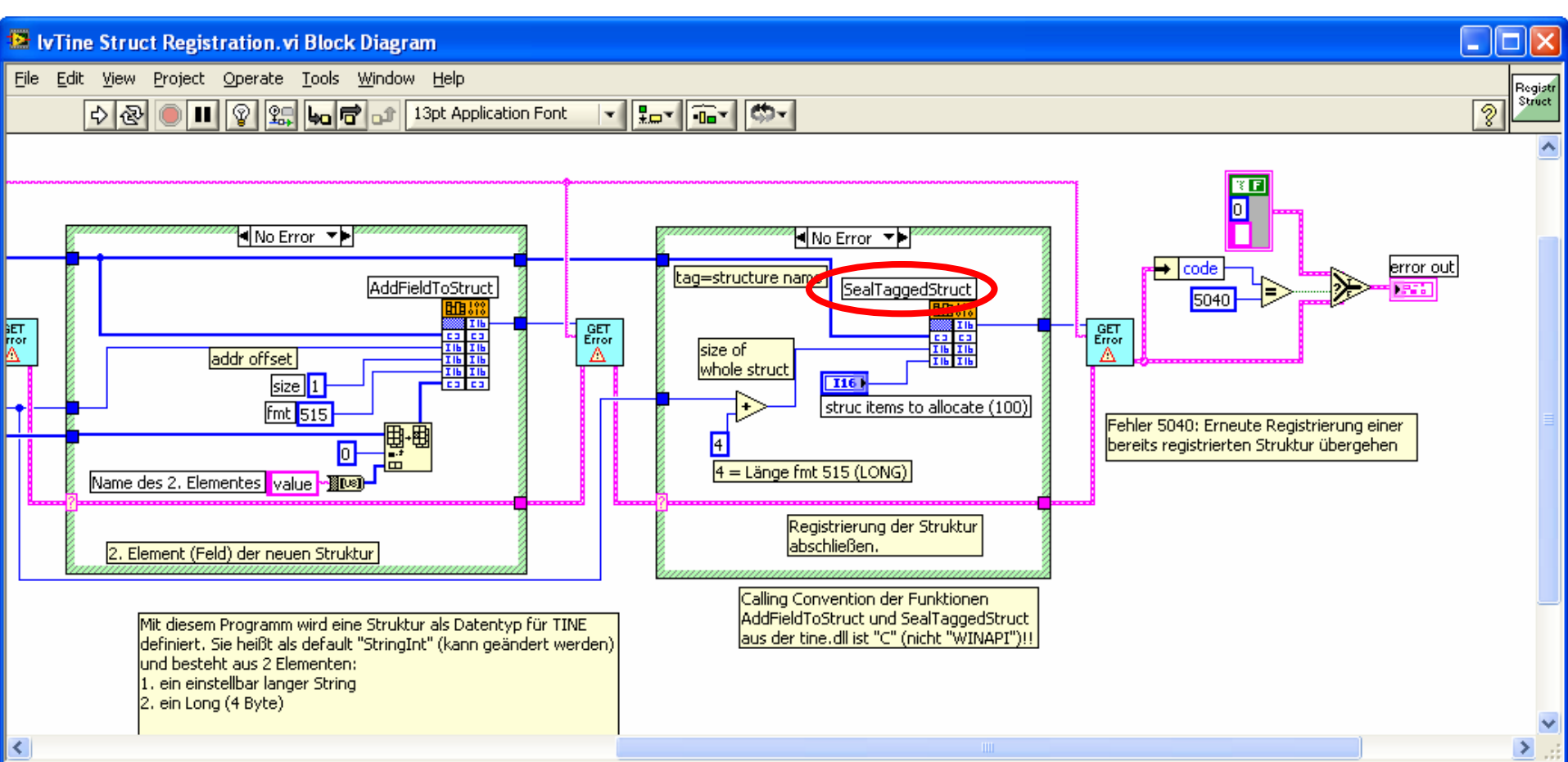


**How we do in LabVIEW**

**Example of register a two element structure:**

1. string (adjustable size)
2. Long (4 bytes)

**AddFieldToStruct** for each element (structure name, offset, size, type, element name) and **SealTaggedStruct** finally (structure name, size of whole struct, structure items to allocate)



# How we do in LabVIEW

- Example of register a two element structure:**
1. string (adjustable size)
  2. Long (4 bytes)

AddFieldToStruct for each element (structure name, offset, size, type, element name) and SealTaggedStruct finally (structure name, size of whole struct, structure items to allocate)

# notice

- definition of structure before server Init!
- structure registration on both sides (srv and cln)

	server	client
out	push	put
in	pull	get