**ENTITY USING DOCS**

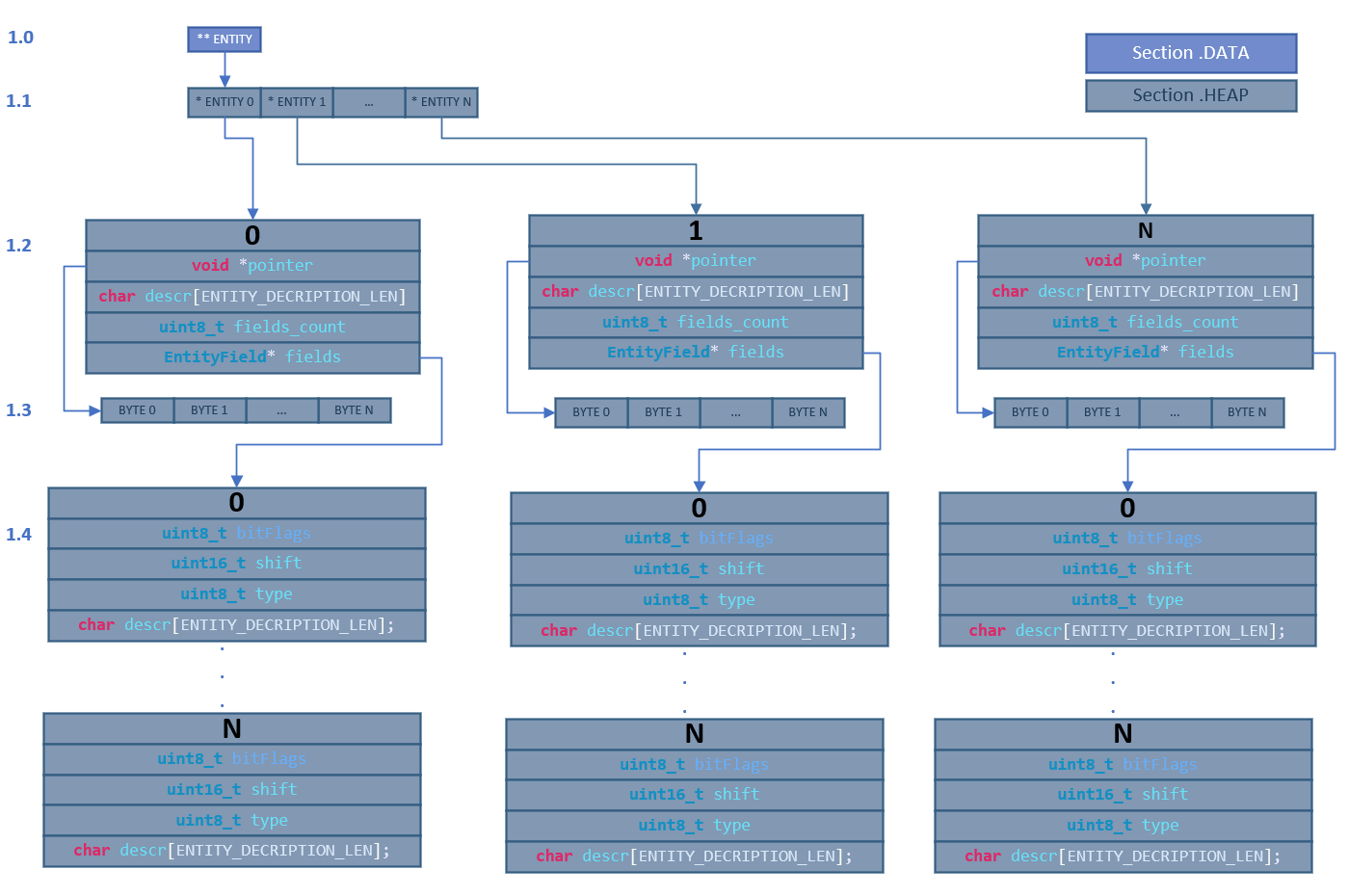


Figure 1. Entity functional diagram.

**uint16\_t** **newEntities**(**uint16\_t** nomberOfEntities); - Clearing all entities and allocating memory for entity pointers *(on Figure 1, section 1.1)*. Function gets quantity of entity pointers. Function returns ENTITY\_OK if allocation is successful and ENTITY\_ERROR when memory is full.

**uint16\_t** **initEntity**(**uint8\_t** NumberOfFields, **uint16\_t** pointerSize, **char** \* descr); - allocating memory for a single entity, also for fields and a pointer to data *(on Figure 1, section 1.2 - 1.4)*. At this stage each member each field entity of *section 1.4 in Figure 1* is initialized to 0. Function gets quantity of fields, data size in byte and entity description. Function returns valid entity number if allocation is successful and ENTITY\_ERROR when memory is full.

**uint8\_t** **initField**(**Entity** \* entityInst, **uint8\_t** \* filedNo, **uint8\_t** readOnly, **uint8\_t** isParameter, **uint8\_t** isLog, **uint16\_t** shift, **uint8\_t** type, **char** \* descr, **void** \* field\_ptr); - initialization allocated field *(on Figure 1, section 1.4).* Function gets pointer on entity, pointer on field number for automatic her counting, flag read only, flag parameter, flag is log, position of pointer, type, description and pointer on value for initialization.

* **flag read only** – tell about this field is read only
* **flag parameter** – if this flag is set then the corresponding field will read from flash before starting system and this field will write to flash if calling corresponding function to writing.
* **flag log -** if this flag is set then the corresponding field will periodic write to flash which log.

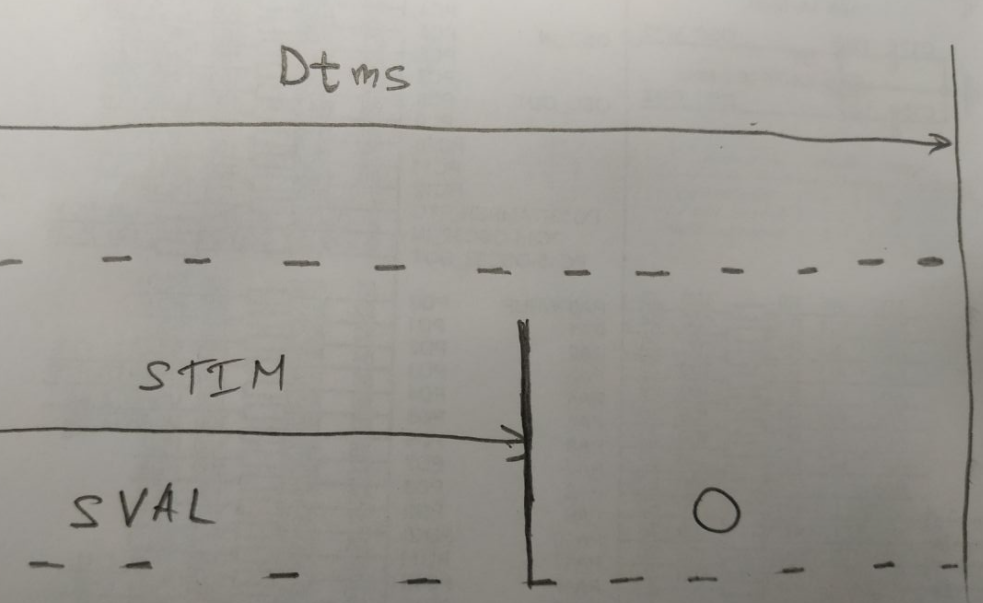
**PID USING DOCS**

In entityPid there are 18 parameters:

* **KP** – float type of proportional coefficient PID
* **KI** – float type of integral coefficient PID
* **KD** – float type of differential coefficient PID
* **P** – proportional component of PID
* **I** – integral component of PID
* **D** – differential component of PID
* **TASK** - supported value
* **PLAN** – current driving value
* **MaxP** – max driving value
* **MinP** – min driving value
* **E** – current error
* **LE** – last error
* **Dtms** – discreteness of PID
* **SVAL** – Hardcode PLAN value in mode 0, 2, 3
* **STIM** – PLAN value holding time in mode 0
* **cOff** – reseting plan time in mode 0, 2, 3
* **ENA** – enable PID
* **MODE** – mode of PID

**PID MODES (Parameter MODE):**

0 - *PULSE\_MANUAL\_MODE*

In this mode PID do not work, parameter **PLAN** is assigned ***SVAL*** value on time ***STIM*** and then, after this time, ***PLAN*** becomes to 0, its repeats of each ***Dtms*** ms. User must will set ***SVAL*, *Dtms*, *STIM,* MODE = 0, ENA = 1**.

1 - STANDART\_PID\_MODE

It`s standart PID mode , in this mode PID changing **PLAN**. User must will set **KP, KI, KD, TASK, *Dtms,* MODE = 1 ENA = 1**parameters.

2- PULSE\_TIME\_ON\_MODE

In this mode PID change **STIM** parameter, ***Dtms = const. PLAN*** value is assigning ***SVAL*** on time **STIM** and then, after this time, ***PLAN*** becomes 0 , its repeats of each ***Dtms*** ms. User must will set **KP, KI, KD, TASK, *Dtms, SVAL,* MODE = 2, ENA = 1**.

3- PULSE\_TIME\_CYCLE\_MODE

In this mode PID change ***Dtms*** parameter, **STIM *= const. PLAN*** value is assigning ***SVAL*** on time **STIM** and then, after this time, ***PLAN*** becomes 0. User must will set **KP, KI, KD, TASK, *Dtms, SVAL,* MODE = 2, ENA = 1**.