Modified

[**midea\_ac\_lan**](https://github.com/georgezhao2010/midea_ac_lan)**/**[**custom\_components**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components)**/**[**midea\_ac\_lan**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan)**/**[**midea**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea)**/**[**devices**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices)**/**[**c3**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices/c3)**/message.py**

|  |  |
| --- | --- |
| from ...core.message import ( | |
|  | | |
| MessageType, | | |  |
|  | MessageRequest, | | |
|  | MessageResponse, | | |
|  | MessageBody, | | |
|  | ) | | |
|  |  | | |
|  |  | | |
|  | class MessageC3Base(MessageRequest): | | |
|  | def \_\_init\_\_(self, device\_protocol\_version, message\_type, body\_type): | | |
|  | super().\_\_init\_\_( | | |
|  | device\_protocol\_version=device\_protocol\_version, | | |
|  | device\_type=0xC3, | | |
|  | message\_type=message\_type, | | |
|  | body\_type=body\_type | | |
|  | ) | | |
|  |  | | |
|  | @property | | |
|  | def \_body(self): | | |
|  | raise NotImplementedError | | |
|  |  | | |
|  |  | | |
|  | class MessageQuery(MessageC3Base): | | |
|  | def \_\_init\_\_(self, device\_protocol\_version): | | |
|  | super().\_\_init\_\_( | | |
|  | device\_protocol\_version=device\_protocol\_version, | | |
|  | message\_type=MessageType.query, | | |
|  | body\_type=0x01) | | |
|  |  | | |
|  | @property | | |
|  | def \_body(self): | | |
|  | return bytearray([]) | | |
|  |  | | |
|  |  | | |
|  | class MessageSet(MessageC3Base): | | |
|  | def \_\_init\_\_(self, device\_protocol\_version): | | |
|  | super().\_\_init\_\_( | | |
|  | device\_protocol\_version=device\_protocol\_version, | | |
|  | message\_type=MessageType.set, | | |
|  | body\_type=0x01) | | |
|  | self.zone1\_power = False | | |
|  | self.zone2\_power = False | | |
|  | self.dhw\_power = False | | |
|  | self.mode = 0 | | |
|  | self.zone\_target\_temp = [25, 25] | | |
|  | self.dhw\_target\_temp = 40 | | |
|  | self.room\_target\_temp = 25 | | |
|  | self.zone1\_curve = False | | |
|  | self.zone2\_curve = False | | |
|  | self.disinfect = False | | |
|  | self.fast\_dhw = False | | |
|  |  | | |
|  | @property | | |
|  | def \_body(self): | | |
|  | # Byte 1 | | |
|  | zone1\_power = 0x01 if self.zone1\_power else 0x00 | | |
|  | zone2\_power = 0x02 if self.zone2\_power else 0x00 | | |
|  | dhw\_power = 0x04 if self.dhw\_power else 0x00 | | |
|  | # Byte 7 | | |
|  | zone1\_curve = 0x01 if self.zone1\_curve else 0x00 | | |
|  | zone2\_curve = 0x02 if self.zone2\_curve else 0x00 | | |
|  | disinfect = 0x04 if self.disinfect else 0x00 | | |
|  | fast\_dhw = 0x08 if self.fast\_dhw else 0x00 | | |
|  | room\_target\_temp = int(self.room\_target\_temp \* 2) | | |
|  | zone1\_target\_temp = int(self.zone\_target\_temp[0]) | | |
|  | zone2\_target\_temp = int(self.zone\_target\_temp[1]) | | |
|  | dhw\_target\_temp = int(self.dhw\_target\_temp) | | |
|  | return bytearray([ | | |
|  | zone1\_power | zone2\_power | dhw\_power, | | |
|  | self.mode, zone1\_target\_temp, zone2\_target\_temp, | | |
|  | dhw\_target\_temp, room\_target\_temp, | | |
|  | zone1\_curve | zone2\_curve | disinfect | fast\_dhw | | |
|  | ]) | | |
|  |  | | |
|  |  | | |
|  | class C3MessageBody(MessageBody): | | |
|  | def \_\_init\_\_(self, body, data\_offset=0): | | |
|  | super().\_\_init\_\_(body) | | |
|  | self.zone1\_power = body[data\_offset + 0] & 0x01 > 0 | | |
|  | self.zone2\_power = body[data\_offset + 0] & 0x02 > 0 | | |
|  | self.dhw\_power = body[data\_offset + 0] & 0x04 > 0 | | |
|  | self.zone1\_curve\_state = body[data\_offset + 0] & 0x08 > 0 | | |
|  | self.zone2\_curve\_state = body[data\_offset + 0] & 0x10 > 0 | | |
|  | self.disinfect = body[data\_offset + 0] & 0x20 > 0 | | |
|  | self.fast\_dhw = body[data\_offset + 0] & 0x40 > 0 | | |
|  | self.zone\_temp\_type = [ | | |
|  | body[data\_offset + 1] & 0x10 > 0, | | |
|  | body[data\_offset + 1] & 0x20 > 0 | | |
|  | ] | | |
|  | self.mode = body[data\_offset + 3] | | |
|  | self.mode\_auto = body[data\_offset + 4] | | |
|  | self.zone\_target\_temp = [ | | |
|  | body[data\_offset + 5], | | |
|  | body[data\_offset + 6] | | |
|  | ] | | |
|  | self.dhw\_target\_temp = body[data\_offset + 7] | | |
|  | self.room\_target\_temp = body[data\_offset + 8] / 2 | | |
|  | self.zone\_heating\_temp\_max = [ | | |
|  | body[data\_offset + 9], | | |
|  | body[data\_offset + 13] | | |
|  | ] | | |
|  | self.zone\_heating\_temp\_min = [ | | |
|  | body[data\_offset + 10], | | |
|  | body[data\_offset + 14] | | |
|  | ] | | |
|  | self.zone\_cooling\_temp\_max = [ | | |
|  | body[data\_offset + 11], | | |
|  | body[data\_offset + 15] | | |
|  | ] | | |
|  | self.zone\_cooling\_temp\_min = [ | | |
|  | body[data\_offset + 12], | | |
|  | body[data\_offset + 16] | | |
|  | ] | | |
|  | self.room\_temp\_max = body[data\_offset + 17] / 2 | | |
|  | self.room\_temp\_min = body[data\_offset + 18] / 2 | | |
|  | self.dhw\_temp\_max = body[data\_offset + 19] | | |
|  | self.dhw\_temp\_min = body[data\_offset + 20] | | |
|  | self.tank\_actual\_temperature = body[data\_offset + 21]  class C3Notify1MessageBody(MessageBody):  def \_\_init\_\_(self, body, data\_offset=0):  super().\_\_init\_\_(body)  status\_byte = body[data\_offset]  self.status\_tbh = bool(status\_byte & 0x08)  self.status\_dhw = bool(status\_byte & 0x04)   |  | | --- | | self.status\_ibh = bool(status\_byte & 0x02) | | self.status\_heating = bool(status\_byte & 0x01) |  | ng = |   self.total\_energy\_consumption = (  (body[data\_offset + 1] << 32) +  (body[data\_offset + 2] << 16) +  (body[data\_offset + 3] << 8) +  (body[data\_offset + 4]))  self.total\_produced\_energy = (  (body[data\_offset + 5] << 32) +  (body[data\_offset + 6] << 16) +  (body[data\_offset + 7] << 8) +  (body[data\_offset + 8]))  # DELETE ALL TOTAL MYSTERY CODE  # TRY THE CODE BELOW  **if body[9] == 0xFF:**  **self.indoor\_temperature = None**  **else:**  **self.indoor\_temperature =** body[data\_offset + 9] / 2  **if body[10] == 0xFF:**  **self.outdoor\_temperature = None**  **else:**  **self.outdoor\_temperature =** body[data\_offset + 10]  self.entering\_water\_temperature = body[data\_offset + 11]  self.leaving\_water\_temperature = body[data\_offset + 12] | | |
|  |  | | |
|  |  | | |
|  | class MessageC3Response(MessageResponse): | | |
|  | def \_\_init\_\_(self, message): | | |
|  | super().\_\_init\_\_(message) | | |
|  | body = message[self.HEADER\_LENGTH: -1] | | |
|  | if (self.\_message\_type in [MessageType.notify1, MessageType.query] and self.\_body\_type == 0x01) or \ | | |
|  | self.\_message\_type == MessageType.notify2: | | |
|  | self.\_body = C3MessageBody(body, data\_offset=1) | | |
|  | **elif self.\_message\_type == MessageType.notify1 and self.\_body\_type == 0x04:**  **self.\_body = C3Notify1MessageBody(body, data\_offset=1)**  self.set\_attr() | | |