ADT en PLC: Cola Circular (FIFO)

Diseño e Implementación en TIA Portal

Piezas de código

Empecemos

Cola Circular (FIFO)

¿Qué es?

✅ Es un ADT que utiliza un array de tamaño fijo.

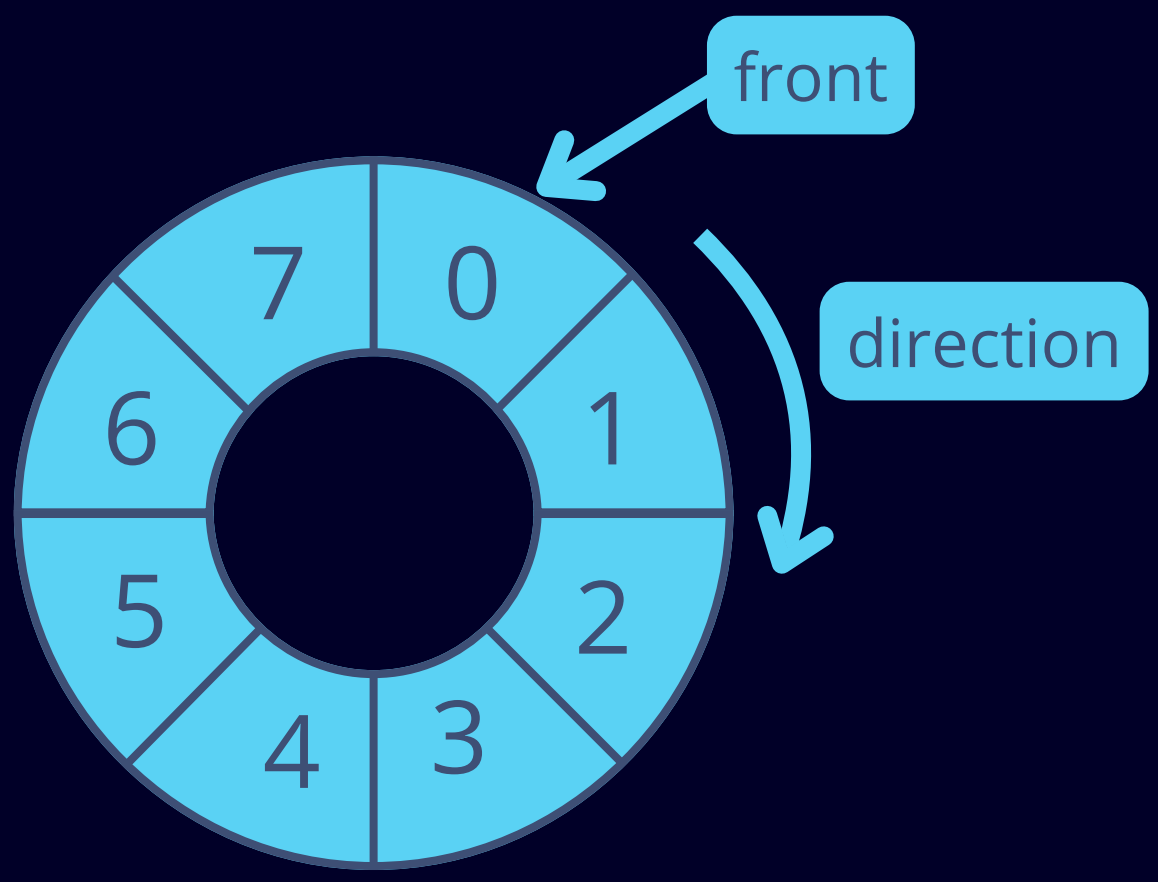
✅ Se comporta como si sus extremos estuvieran conectados formando un círculo

¿Para qué sirve?

✅ Búfer de envío de datos.

✅ Tracking de datos lineal.

✅ Órdenes de producción.



ADT: Abstract Data Types (Tipos Abstractos de Datos)

Atributos (Propiedades) →

Atributos (Propiedades)

Cada instancia de la función "**_queue**" (FC o FB) hace uso de sus atributos/propiedades, la cual es un tipo de dato "**queueInstanceAtributes**" (UDT) donde se estructura como en la siguiente tabla:

ATRIBUTO	TIPO	VALOR INICIAL	VALOR MÍNIMO	VALOR MÁXIMO	DESCRIPCIÓN
front	DINT	0	0	total - 1	Índice del primer elemento de la cola (No siempre coincide con el primer elemento del array)
length	DINT	0	0	total	Número de elementos en la cola
total	DINT	0	0	longitud del array	Cuando se inicializa la cola se calcula el total de elementos del array
isEmpty	BOOL	false	false	true	La cola está vacía
isFull	BOOL	false	false	true	La cola está llena
isInitialized	BOOL	false	false	true	La cola ha sido inicializada con los valores por defecto
mutex	BOOL	false	false	true	Se usa para bloquear el acceso a la cola cuando el ciclo es interrumpido por otro proceso

Abstracción de Datos →

Abstracción de Datos

¡Aquí es donde se “**obra la magia**”!

Imagina que en tu aplicación tienes que encolar dos UDT con estructuras completamente diferentes, sin la abstracción de datos tendrías que crear dos funciones (FC o FB) para gestionar cada UDT.

Gracias al tipo de dato “**Variant**” pasado como parámetro IN_OUT de la función, el PLC es capaz de abstraerse del tipo de dato de la UDT y con eso conseguimos que la función “**_queue**” sea del tipo <E>.

ℹ El tipo <E> o <T> es una convención para representar cualquier tipo genérico. Siendo así, nuestra función cola almacenará datos del tipo <E>, es decir cualquier dato.

Implementación

Para implementar la función (pseudo-clase) he optado por una FC.

¿Por qué? la diferencia entre pasar “instance” por IN_OUT de la FC o guardarla como STATIC de un FB no tiene mucho impacto en el “performance” del bloque. Yo uso FC por simplicidad, usar FB es totalmente válido.

Static	
instance	"queueInstanceAttributes"
data	Array[0..#UPPER_LIMIT] of "userData"

Instancia de la psudo-clase

Array de datos del tipo <E>

Pseudo-método para encolar

Resultado OK

Constant		
UPPER_LIMIT	DInt	7
METHOD_10_ENQUEUE	Int	10
STATUS_0000_DONE	Int	16#0000

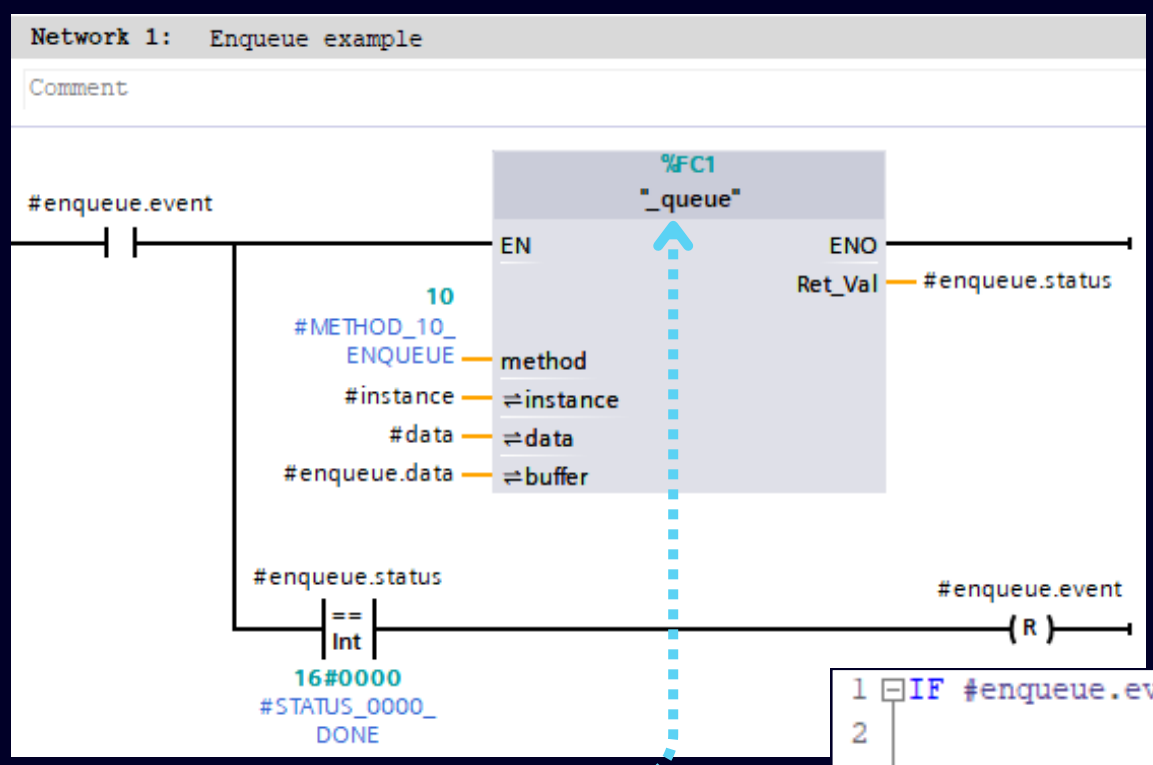
Ejemplo para encolar

Bit que dispara el evento

Datos para encolar

Resultado de la operación

Static	
enqueue	Struct
event	Bool
data	"userData"
status	Int



Ejemplo de implemtación

¡Fácil! Cuando **event** pasa a **TRUE** llama a la función para ejecutar el método **ENQUEUE**, la función devuelve el resultado que se almacena en **status**, evaluamos que **status** sea **DONE** y pasamos **event** a FALSE.

Función síncrona (devuelve resultado en el mismo ciclo de scan)

```
1  IF #enqueue.event THEN
2
3    #enqueue.status := "_queue"(method := #METHOD_10_ENQUEUE,
4                                 instance := #instance,
5                                 data := #data, buffer := #enqueue.data);
6
7    IF #enqueue.status = #STATUS_0000_DONE THEN
8      #enqueue.event := false;
9    END_IF;
10 END_IF;
11
```

Destripando el código →

Destripando el código

Esto es para los obsesos del control...

_queue

		Name	Data type	Default ..	Comment
1		Input			
2		method	Int		1=Init; 10=Enqueue(Add); 11=Dequeue(Remove); 12=Peek
3		Output			
4		<Add new>			
5		InOut			
6		instance	"queueInstanceAttributes"		Queue instance
7		data	Variant		Data array of type <E>
8		buffer	Variant		Buffer data of type <E>
9		Temp			
10		result	Int		
11		length	DInt		
12		rear	DInt		
13		Constant			
14		METHOD_01_INIT	Int	1	Method 01: Initialize
15		METHOD_10_ENQUEUE	Int	10	Method 10: Enqueue (Add) dato to last position
16		METHOD_11_DEQUEUE	Int	11	Method 11: Dequeue (Remove) data from first position
17		METHOD_12_PEEK	Int	12	Method 12: Peek data from first position
18		STATUS_0000_DONE	Int	16#0000	Status: Done
19		STATUS_8002_DATA_IS_NOT_ARRAY	Int	16#8002	Status: Error - Data array is not array
20		STATUS_8003_DATA_IS_NOT_VALID_ARRAY	Int	16#8003	Status: Error - Data array is not valid array
21		STATUS_8005_DATA_ARRAY_LESS_THAN_TWO	Int	16#8005	Status: Error - Data array is less than two elements
22		STATUS_8010_IS_EMPTY	Int	16#8010	Status: Error - ADT is empty
23		STATUS_8011_IS_FULL	Int	16#8011	Status: Error - ADT is full
24		STATUS_8020_ERROR_MOVE_BUFFER_TO_DATA	Int	16#8020	Status: Error - Movement from buffer to data
25		STATUS_8021_ERROR_MOVE_DATA_TO_BUFFER	Int	16#8021	Status: Error - Movement from data to buffer
26		STATUS_8FFF_UNKNOWN_METHOD	Int	16#8FFF	Status: Error - Unknown method
27		Return			
28		_queue	Int		

¿Por qué DINT en arrays, punteros, índices? Porque DINT es el tipo de datos que devuelven muchas funciones de Siemens relacionadas con arrays.

¿Por que INT en status? Porque INT es una convención para devolver resultados básicos

Destripando el código →

Destripando el código

```
43  REGION Method 01: Initialize
44    // 1. Initialize
45    IF NOT #instance.isInitialized OR #method = #METHOD_01_INIT THEN
46
47      // 1.1. Initialize class attributes
48      #instance.front := 0;
49      #instance.length := 0;
50      #instance.total := 0;
51      #instance.isEmpty := 1;
52      #instance.isFull := 0;
53
54      // 1.2. Return if data is not array
55      IF NOT IS_ARRAY(#data) THEN
56        #_queue := #STATUS_8002_DATA_IS_NOT_ARRAY;
57        ENO := FALSE;
58        RETURN;
59      END_IF;
60
61      // 1.3. Get array length
62      #length := UDINT_TO_DINT(CountOfElements(#data));
63
64      // 1.4. Return if array is less than two elements
65      IF #length < 2 THEN
66        #_queue := #STATUS_8005_DATA_ARRAY_LESS_THAN_TWO;
67        ENO := FALSE;
68        RETURN;
69      END_IF;
70
71      // 1.5. Check array limits
72      #result := MOVE_BLK_VARIANT(SRC := #data,
73                                  COUNT := 1,
74                                  SRC_INDEX := 0,
75                                  DEST_INDEX := #length - 1,
76                                  DEST => #data);
77      // 1.6. Return is array is not ok
78      IF #result <> 0 THEN
79        #_queue := #STATUS_8003_DATA_IS_NOT_VALID_ARRAY;
80        ENO := FALSE;
81        RETURN;
82      END_IF;
83
84      // 1.7. Set initialized
85      #instance.isInitialized := TRUE;
86      #instance.total := #length;
87
88      // 1.8. Return done if method was called
89      IF #method = #METHOD_01_INIT THEN
90        #_queue := #STATUS_0000_DONE;
91        ENO := TRUE;
92        RETURN;
93      END_IF;
94    END_IF;
95  END_REGION
```

El pseudo-método: **Init**, si fuera una clase sería el constructor

Inicialización de atributos

Comprobación de tipos

Todo Ok

Destripando el código →

Destripando el código

```
97  CASE #method OF
98    #METHOD_10_ENQUEUE:
99      // ####################################################
00      REGION Method 10: Add to queue
01
02        // 1. Return is queue is full
03        IF #instance.isFull THEN
04          #_queue := #STATUS_8011_IS_FULL;
05          ENO := FALSE;
06          RETURN;
07        END_IF;
08
09        // 2. Calculate rear index
10        #rear := (#instance.front + #instance.length) MOD #instance.total;
11
12        // 3. Move from buffer to data
13        #result := MOVE_BLK_VARIANT(SRC := #buffer,
14                                    COUNT := 1,
15                                    SRC_INDEX := 0,
16                                    DEST_INDEX := #rear,
17                                    DEST => #data);
18
19        // 4. Return if data movement error
20        IF #result <> 0 THEN
21          #_queue := #STATUS_8020_ERROR_MOVE_BUFFER_TO_DATA;
22          ENO := FALSE;
23          RETURN;
24        END_IF;
25
26        // 5. Update class attributes
27        IF #instance.length < #instance.total THEN
28          #instance.length += 1;
29        END_IF;
30
31        #instance.isEmpty := #instance.length <= 0;
32        #instance.isFull := #instance.length >= #instance.total;
33
34        // 6. Return true
35        #_queue := #STATUS_0000_DONE;
36        ENO := true;
37        RETURN;
38      END_REGION ;
39
40    #METHOD_11_DEQUEUE:
41      // ####################################################
42      REGION Method 11: Remove from queue
80
81    #METHOD_12_PEEK:
82      // ####################################################
83      REGION Method 12: Peek first position
11
12  END_CASE;
```

El pseudo-método: **Enqueue**

Comprueba que la cola no esté llena

Movimiento de datos de buffer al array

Actualización de los atributos/propiedades de la instancia

Todo Ok

Destripando el código →

Destripando el código

El pseudo-método: **Dequeue**

```
97  CASE #method OF
98    #METHOD_10_ENQUEUE:
99      // ######################################################
100     REGION Method 10: Add to queue
139
140   #METHOD_11_DEQUEUE:
141     // ######################################################
142     REGION Method 11: Remove from queue
143
144       // 1. Return is queue is empty
145       IF #instance.isEmpty THEN
146         #_queue := #STATUS_8010_IS_EMPTY;
147         ENO := FALSE;
148         RETURN;
149       END_IF;
150
151       // 2. Move from data to buffer
152       #result := MOVE_BLK_VARIANT(SRC := #data,
153                                   COUNT := 1,
154                                   SRC_INDEX := #instance.front,
155                                   DEST_INDEX := 0,
156                                   DEST => #buffer);
157
158     // 3. Return if data movement error
159     IF #result <> 0 THEN
160       #_queue := #STATUS_8021_ERROR_MOVE_DATA_TO_BUFFER;
161       ENO := FALSE;
162       RETURN;
163     END_IF;
164
165     // 4. Update class attributes
166     #instance.front := (#instance.front + 1) MOD #instance.total;
167
168     IF #instance.length > 0 THEN
169       #instance.length -= 1;
170     END_IF;
171
172     #instance.isEmpty := #instance.length <= 0;
173     #instance.isFull := #instance.length >= #instance.total;
174
175     // 5. Return true
176       #_queue := #STATUS_0000_DONE;
177       ENO := true;
178       RETURN;
179     END_REGION ;
180
181   #METHOD_12_PEEK:
182     // ######################################################
183     REGION Method 12: Peek first position
211
212 END_CASE;
```

Comprueba que la cola no esté vacía

Movimiento de datos del array al buffer

Actualización de los atributos/propiedades de la instancia

Todo Ok

Destripando el código →

Destripando el código

El pseudo-método: **Peek**

```
97  CASE #method OF
98    #METHOD_10_ENQUEUE:
99      // ####################################################
100     REGION Method 10: Add to queue
139
140   #METHOD_11_DEQUEUE:
141     // ####################################################
142     REGION Method 11: Remove from queue
180
181   #METHOD_12_PEEK:
182     // ####################################################
183     REGION Method 12: Peek first position
184
185       // 1. Return is queue is empty
186       IF #instance.isEmpty THEN
187         #_queue := #STATUS_8010_IS_EMPTY;
188         ENO := FALSE;
189         RETURN;
190       END_IF;
191
192       // 2. Move from data to buffer
193       #result := MOVE_BLK_VARIANT(SRC := #data,
194                                   COUNT := 1,
195                                   SRC_INDEX := #instance.front,
196                                   DEST_INDEX := 0,
197                                   DEST => #buffer);
198
199       // 3. Return if data movement error
200       IF #result <> 0 THEN
201         #_queue := #STATUS_8021_ERROR_MOVE_DATA_TO_BUFFER;
202         ENO := FALSE;
203         RETURN;
204       END_IF;
205
206       // 4. Return true
207       #_queue := #STATUS_0000_DONE;
208       ENO := true;
209       RETURN;
210     END_REGION ;
211
212 END_CASE;
```

Comprueba que la cola no esté vacía

Movimiento de datos del array al buffer

Todo Ok

Destripando el código →

Destripando el código

```
(*--
# Log
| version | date | author | description |
|:-------:|:----:|:------:|:------------|
| v3.0 | 2025-12-26 | cyanezf | Refactor, not complatible with prev. versions |
| v2.1 | 2024-12-17 | cyanezf | Full refactor and bugs fixed |

# Dependencies
| Dependency |
|:-----------|
| queueInstanceAttributes v3.0 |

# Description
This function is used to queue/dequeue data

## Methods
In any call if queue is not initialized, the function tries to do it.
| Method | Description |
|:-------:|:------------|
| 1 | Initialize queue attributes, that means, function clears queue. |
| 10 | Enqueue (Add) data to last position |
| 11 | Dequeue (Remove) data from first position |
| 12 | Peek data from first position |
| Unknown | Init. is tried and returns an error code. |

## RLO and Return
| RLO | Return | Description |
|:---:|:------:|:------------|
| TRUE | 0 | Done |
| FALSE | 0x8002 | Error - Data array is not array |
| FALSE | 0x8003 | Error - Data array is not valid array |
| FALSE | 0x8005 | Error - Data array is less than two elements |
| FALSE | 0x8010 | Error - ADT is empty |
| FALSE | 0x8011 | Error - ADT is full |
| FALSE | 0x8020 | Error - Movement from buffer to data |
| FALSE | 0x8021 | Error - Movement from data to buffer |
| FALSE | 0x8FFF | Error - Unknown method |

#
_Use [Markdown Live Preview](https://markdownlivepreview.com/) to watch this doc._
--*)

REGION Method 01: Initialize

CASE #method OF ... END_CASE;

  // Return
  #_queue := #STATUS_8FFF_UNKNOWN_METHOD;
  ENO := FALSE;
```

Control de versiones y documentación del código

Método desconocido

Para finalizar

¿Y tú?

¿Cómo implementas tus colas (FIFOS)?

¿Eres team FC o FB?

👇 ¡Nos vemos en los comentarios!

#PLC #ADT #Queue #FIFO

🏅 Fin