message\_send -> message\_receive

FREERTOS:

1. xQueueSendToBack(tm\_queue\_array[queue\_id], (const void\*) message\_ptr, (TickType\_t) 0)

* xQueueGenericSend()

1. Enter Critical()
2. prvCopyDataToQueue( pxQueue, pvItemToQueue, xCopyPosition );
3. memcpy( ( void \* ) pxQueue->pcWriteTo, pvItemToQueue, ( size\_t ) pxQueue->uxItemSize );
4. Exit Critical()
5. xQueueReceive()
6. Enter Critical()
7. prvCopyDataFromQueue( pxQueue, pvBuffer );
8. memcpy( ( void \* ) pvBuffer, ( void \* ) pxQueue->u.xQueue.pcReadFrom, ( size\_t ) pxQueue->uxItemSize );
9. Exit Critical()

message queue empty:

1. xQueueReceive()
2. Enter Critical()
3. ticks to wait = 0 t=> Exit Critical()

Bemerkungen:

Initializing FreeRTOS system...

Starting Thread-Metric tests...

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Mit einem thread\_sleep(1) bevor die queue gefüllt wird.

Total messages processed: 1

Total interrupts processed: 1

Profile Point: msg\_latency

Cycle Count: 3075

ICache Miss Count: 17

DCache Access Count: 455

DCache Miss Count: 2

Initializing FreeRTOS system...

Starting Thread-Metric tests...

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Total messages processed: 1

Total interrupts processed: 1

Profile Point: msg\_latency

Ohne den thread\_sleep(1) bevor die queue gefüllt wird.

Cycle Count: 32680

ICache Miss Count: 53

DCache Access Count: 7287

DCache Miss Count: 31

**ThreadX:**

1. tx\_queue\_send(&tm\_queue\_array[queue\_id], message\_ptr, TX\_NO\_WAIT)
2. TX\_DiSABLE
3. TX\_QUEUE\_MESSAGE\_COPY(source, destination, size) // 2 times
4. TX\_RESTORE
5. tx\_queue\_receive(TX\_QUEUE \*queue\_ptr, VOID \*destination\_ptr, ULONG wait\_option)
6. TX\_DISABLE
7. TX\_QUEUE\_MESSAGE\_COPY(source, destination, size) // 2 times
8. TX\_RESTORE

message queue empty:

1. tx\_queue\_receive
2. TX\_DISABLE
3. TX\_RESTORE

**Beobachtungen:**

Initializing ThreadX system...

Starting Main Thread...

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Total messages processed: 1

Total interrupts processed: 1

Mit einem thread\_sleep(1) bevor die queue gefüllt wird.

Profile Point: msg\_latency

Cycle Count: 3200

ICache Miss Count: 36

DCache Access Count: 285

DCache Miss Count: 0

Initializing ThreadX system...

Starting Main Thread...

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Total messages processed: 1

Ohne den thread\_sleep(1) bevor die queue gefüllt wird.

Total interrupts processed: 1

Profile Point: msg\_latency

Cycle Count: 3676

ICache Miss Count: 44

DCache Access Count: 311

DCache Miss Count: 1

**Zephyr:**

1. **z\_impl\_k\_msgq\_put**(**struct** k\_msgq \*msgq, **const** **void** \*data, k\_timeout\_t timeout)
2. k\_spin\_lock(&msgq->lock); // Interrupts deaktivieren
3. (**void**)**memcpy**(msgq->write\_ptr, (**char** \*)data, msgq->msg\_size);
4. z\_reschedule(&msgq->lock, key); // Interrupts deaktivieren
5. **z\_impl\_k\_msgq\_get**(**struct** k\_msgq \*msgq, **void** \*data, k\_timeout\_t timeout)
6. k\_spin\_lock(&msgq->lock)
7. (**void**)**memcpy**((**char** \*)data, msgq->read\_ptr, msgq->msg\_size);
8. k\_spin\_unlock(&msgq->lock, key);

with empty message queue:

1. k\_spin\_lock(&msgq->lock);
2. k\_spin\_unlock(&msgq->lock, key);

**Beobachtungen:**

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Total messages processed: 1

Mit einem thread\_sleep(1) bevor die queue gefüllt wird.

Total interrupts processed: 1

Profile Entry: msg\_latency

Cycle Count: 1122

[ICache Miss]: 11

[DCache Access]: 148

[DCache Miss]: 0

[Init] ISR-to-Task Message Queue Benchmark started.

==== OneShot Benchmark Complete ====

Total messages processed: 1

Total interrupts processed: 1

Ohne den thread\_sleep(1) bevor die queue gefüllt wird.

Profile Entry: msg\_latency

Cycle Count: 1398

[ICache Miss]: 16

[DCache Access]: 154

[DCache Miss]: 5